Joining Hands: Exploiting Monolingual Treebanks for Parsing of Code-mixing Data

Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Manish Shrivastava and Dipti Misra Sharma

LTRC, IIIT-H, Hyderabad, India

{irshad.bhat,riyaz.bhat,m.shrivastava,dipti}@iiit.ac.in

Abstract

In this paper, we propose efficient and less resource-intensive strategies for parsing of code-mixed data. These strategies are not constrained by in-domain annotations, rather they leverage pre-existing monolingual annotated resources for training. We show that these methods can produce significantly better results as compared to an informed baseline. Besides, we also present a data set of 450 Hindi and English code-mixed tweets of Hindi multilingual speakers for evaluation. The data set is manually annotated with Universal Dependencies.

1 Introduction

Code-switching or code-mixing is a sociolinguistic phenomenon, where multilingual speakers switch back and forth between two or more common languages or language varieties in a single utterance[1]. The phenomenon is mostly prevalent in spoken language and in informal settings on social media such as in news groups, blogs, chat forums etc. Computational modeling of code-mixed data, particularly from social media, is presumed to be more challenging than monolingual data due to various factors. The main contributing factors are non-adherence to a standard grammar, spelling variations and/or back-transliteration. It has been generally observed that traditional NLP techniques perform miserably when processing code-mixed language data (Solorio and Liu, 2008b; Vyas et al., 2014; Çetinoğlu et al., 2016).

[1]For brevity, we will not differentiate between intra- and inter-sentential mixing of languages and use the terms code-mixing and code-switching interchangeably throughout the paper.

More recently, there has been a surge in studies concerning code-mixed data from social media (Solorio and Liu, 2008a; Solorio and Liu, 2008a; Vyas et al., 2014; Sharma et al., 2016; Rudra et al., 2016; Joshi et al., 2016, and others). Besides these individual research articles, a series of shared-tasks and workshops on preprocessing and shallow syntactic analysis of code-mixed data have also been conducted at multiple venues such as Empirical Methods in NLP (EMNLP 2014 and 2016), International Conference on NLP (ICON 2015 and 2016) and Forum for Information Retrieval Evaluation (FIRE 2015 and 2016). Most of these works are an attempt to address preprocessing issues– such as language identification and transliteration– that any higher NLP application may face in processing such data.

Due to paucity of annotated resources in code-mixed genre, the performance of monolingual parsing models is yet to be evaluated on code-mixed structures. This paper serves to fill this gap by presenting an evaluation set annotated with dependency structures. Besides, we also propose different parsing strategies that exploit nothing but the pre-existing annotated monolingual data. We show that by making trivial adaptations, monolingual parsing models can effectively parse code-mixed data.

2 Parsing Strategies

We explore three different parsing strategies to parse code-mixed data and evaluate their performance on a manually annotated evaluation set. These strategies are distinguished by the way they use pre-existing treebanks for parsing code-mixed data.

- **Monolingual:** The monolingual method uses two separate models trained from the respective

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 324–330, Valencia, Spain, April 3-7, 2017.

monolingual treebanks of the languages which are present in the code-mixed data. We can use the monolingual models in two different ways. Firstly, we can parse each code-mixed sentence by intelligently choosing the monolingual model based on the matrix language of the sentence.[2] A clear disadvantage of this method is that the monolingual parser may not accurately parse those fragments of a sentence which belong to a language unknown to the model. Therefore, we consider this as the baseline method. Secondly, we can linearly interpolate the predictions of both monolingual models at the inference time. The interpolation weights are chosen based on the matrix language of each parsing configuration. The interpolated oracle output is defined as:

$$y = \text{argmax}(\lambda_m * f(\phi(c_m)) + (1 - \lambda_m) * f(\phi(c_s))) \quad (1)$$

where $f(\cdot)$ is a *softmax* layer of our neural parsing model, $\phi(c_m)$ and $\phi(c_s)$ are the feature functions of the matrix and subordinate languages respectively and λ_m is the interpolation weight for the matrix language (see Section §5 for more details on the parsing model).

Instead of selecting the matrix language at sentence level, we define the matrix language individually for each parsing configuration. We define the matrix language of a configuration based on the language tags of top 2 nodes in the stack and buffer belonging to certain syntactic categories such as adposition, auxiliary, particle and verb.

- **Multilingual:** In the second approach, we train a single model on a combined treebank of the languages represented in the code-mixed data. This method has a clear advantage over the baseline Monolingual method in that it would be aware of the grammars of both languages of the code-mixed data. However, it may not be able to properly connect the fragments of two languages as the model lacks evidence for such mixed structures in the augmented data. This would particularly happen if the code-mixed languages are typologically diverse. Moreover, training a parsing model on augmented data with more diverse structures will worsen the structural ambiguity problem. But we can easily circumvent this problem by including token-level language tag as an additional feature in the parsing model (Ammar et al., 2016).
- **Multipass:** In the Multipass method, we train two separate models like the Monolingual method. However, we apply these models on the code-mixed data differently. Unlike Monolingual method, we use both models simultaneously for each sentence and pass the input to the models twice. There are two possible ways to accomplish this. We can first parse all the fragments of each language using their respective parsing models one by one and then the root nodes of the parsed fragments would be parsed by the matrix language parsing model. Or, we can parse the subordinate language first and then parse the root of the subordinate fragments with the fragments of matrix language using the matrix language parser. In both cases, monolingual parsers would not be affected by the cross language structures. More importantly, matrix language parser in the second pass would be unaffected by the internal structure of the subordinate language fragments. But there is a caveat, we need to identify the code-mixed fragments accurately, which is a non-trivial task. In this paper, we use token-level language information to segment tweets into subordinate or matrix language fragments.

3 Code-mixed Dependency Annotations

To the best of our knowledge, there is no available code-mixed data set that contains dependency annotations. There are, however, a few available code-mixed data sets that provide annotations related to language of a token, its POS and chunk tags. For an intrinsic evaluation of our parsing models on code-mixed texts, we manually annotated a data set of Hindi-English code-mixed tweets with dependency structures. The code-mixed tweets were sampled from a large set of tweets of Indian language users that we crawled from Twitter using Tweepy[3]–a Twitter API wrapper. We used a language identification system (see §4) to filter Hindi-English code-mixed tweets from the crawled Twitter data. Only those tweets

[2] In any code-mixed utterance, the matrix language defines the overall grammatical structure of an utterance, while subordinate language represents any individual words or phrases embedded in the matrix language. We use a simple count-based approach to identify the matrix and subordinate languages of a code-mixed sentence.

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

were selected that satisfied a minimum ratio of 30:70(%) code-mixing. From this data set, we manually selected 450 tweets for annotation. The selected tweets are thoroughly checked for code-mixing ratio. While calculating the code-mixing ratio, we do not consider borrowings from English as an instance of code-mixing. For POS tagging and dependency annotation, we used Universal dependency guidelines (De Marneffe et al., 2014), while language tags are assigned based on the tagset defined in (Solorio et al., 2014; Jamatia et al., 2015). The annotations are split into testing and tuning sets for evaluation and tuning of our models. The tuning set consists of 225 tweets (3,467 tokens) with a *mixing ratio* of 0.54 and the testing set contains 225 tweets (3,322 tokens) with a *mixing ratio* of 0.53. Here *mixing ratio* is defined as:

$$\frac{1}{n}\sum_{s=1}^{n}\frac{H_s}{H_s + E_s} \quad (2)$$

where n is the number of sentences in the data set, H_s and E_s are the number of Hindi words and English words in sentence s respectively.

4 Preprocessing

The parsing strategies that we discussed above for code-mixed texts heavily rely on language identification of individual tokens. Besides we also need normalization of non-standard word forms prevalent in code-mixed social media content and back-transliteration of Romanized Hindi words. Here we discuss both preprocessing steps in brief.

Language Identification We model language identification as a classification problem where each token needs to be classified into one of the following tags: 'Hindi' (hi), 'English' (en), 'Acronym' (acro), 'Named Entity' (ne) and 'Universal' (univ). For this task, we use the feed-forward neural network architecture of Bhat et al. (2016)[4] proposed for Named Entity extraction in code mixed-data of Indian languages. We train the network with similar feature representations on the data set provided in ICON 2015[5] shared task on language identification. The data set contains 728 Facebook comments annotated with the five language tags noted above. We evaluated the predictions of our identification system against the gold language tags in our code-mixed development set and test set. Even though the model is trained on a very small data set, its prediction accuracy is still above 96% for both the development set and the test set. The results are shown in Table 1.

Normalization and Transliteration We model the problem of both normalization and back-transliteration of (noisy) Romanized Hindi words as a single transliteration problem. Our goal is to learn a mapping for both standard and non-standard Romanized Hindi word forms to their respective standard forms in Devanagari. For this purpose, we use the structured perceptron of Collins (Collins, 2002) which optimizes a given loss function over the entire observation sequence. For training the model, we use the transliteration pairs (87,520) from the Libindic transliteration project[6] and Brahmi-Net (Kunchukuttan et al., 2015) and augmented them with noisy transliteration pairs (63,554) which are synthetically generated by dropping non-initial vowels and replacing consonants based on their phonological proximity. We use Giza++ (Och and Ney, 2003) to character align the transliteration pairs for training.

At inference time, our transliteration model would predict the most likely word form for each input word. However, the single-best output from the model may not always be the best option considering an overall sentential context. Contracted word forms in social media content are quite often ambiguous and can represent different standard word forms such as 'pt' may refer to 'put', 'pit', 'pat', 'pot' and 'pet'. To resolve this ambiguity, we extract n-best transliterations from the transliteration model using beam-search decoding. The best word sequence is then decoded using an exact search over b^n word sequences[7] scored by a tri-gram language model. The language model is trained on monolingual data using IRSTLM-Toolkit (Federico et al., 2008) with Kneser-Ney smoothing. For English, we use a similar model for normalization which we trained on the noisy word forms (3,90,000) synthetically generated from the English vocabulary.

[4]Due to space limitation we don't discuss the system architecture in detail. The interested reader can refer to the original paper for a detailed description.

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

[6]https://github.com/libindic/indic-trans

[7]b is the size of beam-width and n is the sentence length. For each word, we extract five best transliterations or normalizations i.e., b=5.

Label	Development-Set				Test-Set			
	Precision	Recall	F1-Score	Count	Precision	Recall	F1-Score	Count
acro	0.920	0.742	0.821	31	0.955	0.724	0.824	29
en	0.962	0.983	0.972	1303	0.952	0.981	0.966	1290
hi	0.971	0.975	0.973	1545	0.968	0.964	0.966	1460
ne	0.915	0.701	0.794	154	0.889	0.719	0.795	167
univ	0.982	0.995	0.989	434	0.987	1.000	0.993	376
Accuracy	0.967			3467	0.961			3322

Table 1: Language Identification results on code-mixed development set and test set.

5 Experimental Setup

The parsing experiments reported in this paper are conducted using a non-linear neural network-based transition system which is similar to (Chen and Manning, 2014). The models are trained on Universal Dependency Treebanks of Hindi and English released under version 1.4 of Universal Dependencies (Nivre et al., 2016).

Parsing Models Our parsing model is based on transition-based dependency parsing paradigm (Nivre, 2008). Particularly, we use an arc-eager transition system (Nivre, 2003). The arc-eager system defines a set of configurations for a sentence w_1,...,w_n, where each configuration C = (S, B, A) consists of a stack S, a buffer B, and a set of dependency arcs A. For each sentence, the parser starts with an initial configuration where S = [ROOT], B = [w_1,...,w_n] and $A = \emptyset$ and terminates with a configuration C if the buffer is empty and the stack contains the ROOT. The parse trees derived from transition sequences are given by A. To derive the parse tree, the arc-eager system defines four types of transitions (t): 1) Shift, 2) Left-Arc, 3) Right-Arc, and 4) Reduce.

Similar to (Chen and Manning, 2014), we use a non-linear neural network to predict the transitions for the parser configurations. The neural network model is the standard feed-forward neural network with a single layer of hidden units. We use 200 hidden units and RelU activation function. The output layer uses softmax function for probabilistic multi-class classification. The model is trained by minimizing cross entropy loss with an $l2$-regularization over the entire training data. We also use mini-batch Adagrad for optimization (Duchi et al., 2011) and apply dropout (Hinton et al., 2012).

From each parser configuration, we extract features related to the top four nodes in the stack, top four nodes in the buffer and leftmost and rightmost children of the top two nodes in the stack and the leftmost child of the top node in the buffer.

POS Models We train POS tagging models using a similar neural network architecture as discussed above. Unlike (Collobert et al., 2011), we do not learn separate transition parameters. Instead we include the structural features in the input layer of our model with other lexical and non-lexical units. We use second-order structural features, two words to either side of the current word, and last three characters of the current word.

We trained two POS tagging models: *Monolingual* and *Multilingual*. In the Monolingual approach, we divide each code-mixed sentence into contiguous fragments based on the language tags assigned by the language identifier. Words with language tags other than 'Hi' and 'En' (such as univ, ne and acro) are merged with the preceding fragment. Each fragment is then individually tagged by the monolingual POS taggers trained on their respective monolingual POS data sets. In the Multilingual approach, we train a single model on combined data sets of the languages in the code-mixed data. We concatenate an additional 1x2 vector[8] in the input layer of the neural network representing the language tag of the current word. Table 2 gives the POS tagging accuracies of the two models.

Model	LID	Development-Set			Test-Set		
		HIN	ENG	Total	HIN	ENG	Total
Monolingual	G	0.849	0.903	0.873	0.832	0.889	0.860
	A	0.841	0.892	0.866	0.825	0.883	0.853
Multilingual	G	0.835	0.903	0.867	0.798	0.892	0.843
	A	0.830	0.900	0.862	0.790	0.888	0.836

Table 2: POS Tagging accuracies for monolingual and multilingual models. LID = Language tag, G = Gold LID, A = Auto LID.

Word Representations For both POS tagging and parsing models, we include the lexical features in the input layer of the Neural Network using the pre-trained word representations while for the non-lexical features, we use randomly initialized embeddings within a range of -0.25 to $+0.25$.[9] We use Hindi and English monolingual corpora to learn the distributed representation of the lexical units. The English monolingual data contains around 280M sentences, while the Hindi data is comparatively smaller and contains around 40M sentences. The word representations are learned using Skip-gram model with negative sampling which is implemented in `word2vec` toolkit (Mikolov et al., 2013). For multilingual models, we use robust projection algorithm of Guo et al. (2015) to induce bilingual representations

[8] In our experiments we fixed these to be {-0.25,0.25} for Hindi and {0.25,-0.25 } for English

[9] Dimensionality of input units in POS and parsing models: 80 for words, 20 for POS tags, 2 for language tags and 20 for affixes.

Data-set	Gold (POS + language tag)										Auto (POS + language tag)									
	Monolingual		**Interpolated**		**Multilingual**		**Multipass$_f$**		**Multipass$_s$**		**Monolingual**		**Interpolated**		**Multilingual**		**Multipass$_f$**		**Multipass$_s$**	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
CM$_d$	60.77	49.24	74.62	64.11	**75.77**	**65.32**	69.37	58.83	70.23	59.64	55.80	43.36	**68.24**	**56.07**	67.71	55.18	63.34	52.22	64.60	53.03
CM$_t$	60.05	48.52	**74.40**	63.65	74.16	**64.11**	68.54	57.87	69.12	58.64	54.95	43.03	65.14	54.00	**66.18**	**54.40**	62.37	51.11	63.74	52.34
HIN$_t$	**93.29**	**90.60**	92.61	89.64	91.96	88.46	**93.29**	**90.60**	**93.29**	**90.60**	**91.92**	**88.39**	91.82	88.34	89.52	84.83	**91.92**	**88.39**	**91.92**	**88.39**
ENG$_t$	85.12	**82.86**	84.21	81.82	**85.16**	82.79	85.12	**82.86**	85.12	**82.86**	**83.28**	**79.90**	82.08	78.54	82.53	79.11	**83.28**	**79.90**	**83.28**	**79.90**

Table 3: Accuracy of different parsing strategies on Code-mixed as well as Hindi and English evaluation sets. CM$_{d|t}$ = Code-mixed development and testing sets; HIN$_t$ = Hindi test set; ENG$_t$ = English test set; Multipass$_{f|s}$ = fragment-wise and subordinate-first parsing methods.

using the monolingual embedding space of English and a bilingual lexicon of Hindi and English (∼63,000 entries). We extracted the bilingual lexicon from ILCI and Bojar Hi-En parallel corpora (Jha, 2010; Bojar et al., 2014).

6 Experiments and Results

We conducted multiple experiments to measure effectiveness of the proposed parsing strategies in both gold and predicted settings. In predicted settings, we use the monolingual POS taggers for all the experiments. We used the Monolingual method as the baseline for evaluating other parsing strategies. The baseline model parses each sentence in the evaluation sets by either using Hindi or English parsing model based on the matrix language of the sentence. For baseline and the Multipass methods, we use bilingual embedding space derived from matrix language embedding space (Hindi or English) to represent lexical nodes in the input layer of our parsing architecture. In the Interpolation method, we use separate monolingual embedding spaces for each model. The interpolation weights are tuned using the development set and the best results are achieved at λ_m ranging from 0.7 to 0.8 (see eq. 1). The results of our experiments are reported in Table 3. Table 4 shows the impact of sentential decoding for choosing the best normalized and/or back-transliterated tweets on different parsing strategies (see §4).

Data-set	First Best				K-Best			
	Multilingual		**Interpolated**		**Multilingual**		**Interpolated**	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
CM$_d$	66.21	53.55	66.70	53.68	67.71	55.18	68.24	56.07
CM$_t$	65.87	53.92	64.26	53.35	66.18	54.40	65.14	54.00

Table 4: Parsing accuracies with exact search and k-best search (k = 5). CM$_{d|t}$ = Code-mixed development and testing sets.

All of our parsing models produce results that are at-least 10 LAS points better than our baseline parsers which otherwise provide competitive results on Hindi and English evaluation sets (Straka et al., 2016).[10] Among all the parsing strategies, the Interpolated methods perform comparatively better on both monolingual and code-mixed evaluation sets. Interpolation method manipulates the parameters of both languages quite intelligently at each parsing configuration. Despite being quite accurate on code-mixed evaluation sets, the Multilingual model is less accurate in single language scenario. Also the Multilingual model performs worse for Hindi since its lexical representation is derived from English embedding space. It is at-least 2 LAS points worse than the Interpolated and the Multipass methods. However, unlike the latter methods, the Multilingual models do not have a run-time and computational overhead. In comparison to Interpolated and Multilingual methods, Multipass methods are mostly affected by the errors in language identification. Quite often these errors lead to wrong segmentation of code-mixed fragments which adversely alter their internal structure.

Despite higher gains over the baseline models, the performance of our models is nowhere near the performance of monolingual parsers on newswire texts. This is due to inherent complexities of code-mixed social media content (Solorio and Liu, 2008b; Vyas et al., 2014; Çetinoğlu et al., 2016).

7 Conclusion

In this paper, we have evaluated different strategies for parsing code-mixed data that only leverage monolingual annotated data. We have shown that code-mixed texts can be efficiently parsed by the monolingual parsing models if they are intelligently manipulated. Against an informed monolingual baseline, our parsing strategies are at-least 10 LAS points better. Among different strategies that we proposed, Multilingual and Interpolation methods are two competitive methods for parsing code-mixed data.

The code of the parsing models is available at the GitHub repository `https://github.com/irshadbhat/cm-parser`, while the data can be found under the Universal Dependencies of Hindi at `https://github.com/UniversalDependencies/UD_Hindi`.

[10] Our results are not directly comparable to (Straka et al., 2016) due to different parsing architectures. While we use a simple greedy, projective transition system, Straka et al. (2016) use a search-based swap system.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Irshad Ahmad Bhat, Manish Shrivastava, and Riyaz Ahmad Bhat. 2016. Code mixed entity extraction in indian languages using neural networks. In *Proceedings of the Shared Task on Code Mix Entity Extraction in Indian Languages (CMEE-IL)*.

Ondřej Bojar, Vojtěch Diatka, Pavel Rychlý, Pavel Straňák, Vít Suchomel, Aleš Tamchyna, and Daniel Zeman. 2014. HindEnCorp - Hindi-English and Hindi-only Corpus for Machine Translation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Özlem Çetinoğlu, Sarah Schulz, and Ngoc Thang Vu. 2016. Challenges of computational processing of code-switching. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 1–11, Austin, Texas, November. Association for Computational Linguistics.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, volume 14, pages 4585–92.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul).

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. Irstlm: an open source toolkit for handling large scale language models. In *Interspeech*, pages 1618–1621.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1234–1244.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2015. Part-of-speech tagging for code-mixed english-hindi twitter and facebook chat messages. page 239.

Girish Nath Jha. 2010. The TDIL program and the Indian language corpora initiative (ILCI). In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC 2010). European Language Resources Association (ELRA)*.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Anoop Kunchukuttan, Ratish Puduppully, and Pushpak Bhattacharyya. 2015. Brahmi-net: A transliteration and script conversion system for languages of the indian subcontinent.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Eckhard Bick, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Fabricio Chalub, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Claudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh

Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Linh Hà M, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lucia Lam, Phng Lê Hng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Robert Östling, Lilja Øvrelid, Valeria Paiva, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Mats Wirén, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

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

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Shrivastava, Radhika Mamidi, and Dipti M. Sharma. 2016. Shallow parsing pipeline - hindi-english code-mixed social media text. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1340–1345, San Diego, California, June. Association for Computational Linguistics.

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

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

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

Milan Straka, Jan Hajic, and Jana Straková. 2016. Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia*, pages 4290–4297.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 14, pages 974–979.