Attention based Anaphora Resolution for Code-Mixed Social Media Text for Hindi Language

Sandhya Singh, Kevin Patel and Pushpak Bhattacharyya

Center for Indian Language Technology (CFILT), IIT Bombay, Mumbai, India

Abstract

Anaphora resolution is a challenging problem in Natural Language Processing. Its complexity is further aggravated when applied in a social media setting, often due to inherent challenges in code-mixed data widely prevalent in such a setting. In this paper, we investigate the problem of anaphora resolution for code-mixed (Hindi and English) Twitter data. We propose a modified encoder-decoder with attention model for anaphora resolution. Results of our preliminary investigations indicate that such attention based approaches are indeed useful for this task, and should be explored further.

Keywords

Anaphora Resolution, Code-Mixed, Tweets, Hindi anaphora, Encoder-Decoder,

1. Introduction

Anaphora is a linguistic expression used to refer back to an entity to avoid repetition in text. The process of identifying the entity which is being referred by the anaphora in consideration is known as anaphora resolution. It is an important component of Natural Language Processing (NLP) pipeline [1]. Anaphora resolution is necessary for multiple NLP tasks like Information Extraction, Summarization, Question-Answering *etc.* It is a known challenging problem in NLP. As per Winograd schema challenge[1], which is acknowledged as an alternative to Turing test[2] for machine intelligence, solving anaphora resolution task is a benchmark to true machine intelligence.

In current times, social media has become an integral platform for text communication globally. With the participants joining from diverse multi-lingual backgrounds, mixing languages in text communication have become a common phenomenon. The mixing of phrases, words and morphemes of one language into another language is referred as code-mixing. The code-mixing can be at inter-sentential, intra-sentential and intra-word levels. Figure 1 example shows a sample of code-mixed data with code-mixing at inter-sentential, intra-sentential and intra-word boundary. The example text is neither a complete Hindi text nor English language text with spelling abbreviation for *"great"* and *"lifetime"*. With emoticons and abbreviations mixed in

FIRE 2020: Forum for Information Retrieval Evaluation, December 16-20, 2020, Hyderabad, India

sandhyasingh@cse.iitb.ac.in (S. Singh); kevin.patel@cse.iitb.ac.in (K. Patel); pb@cse.iitb.ac.in (P. Bhattacharyya)

[1] http://commonsensereasoning.org/winograd.html

[2] https://en.wikipedia.org/wiki/Turing_test

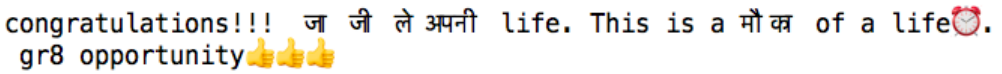

Figure 1: Code-mixed Tweet example

the text, its challenging for the machine to process it in current state. This code-mixed data is a relevant resource for addressing different social media related Natural Language Processing (NLP) problems. The code mixing has added new dimensions to the already existing challenges of NLP.

The challenges posed by code-mixed data are spelling errors, creative spelling variations, abbreviations, transliterations, hybrid grammar, use of emoticons, use of meta tags and hash tags etc [2, 3]. In addition to these, social media platforms also induce size limitations in communication. So the social media code-mixed data, in general, is very short in nature. This induces the challenge of insufficient context to find the antecedent of an anaphor without any background knowledge. Sometimes the text is in reference to the context mentioned in previous posts which were posted hours before or it refers to some world knowledge entity.

Consider the example in Figure 2. Here, the social media text is primarily in Hindi language with the phrase "batting order" borrowed from English. Also, "usake" is a third-person pronoun in Hindi, referring to real-world entity "Virat Kohli", the captain of the Indian cricket team. To resolve the antecedent of this tweet, timeline of the tweet and world knowledge is required. All these challenges have added to already existing challenges of anaphora resolution task.

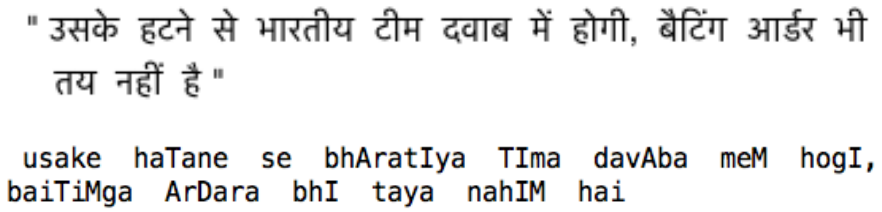

Figure 2: Tweet example

In this paper, we are attempting to address the problem of code-mixed anaphora resolution task for Hindi language using a deep learning based encoder-decoder model with attention. The rest of the paper is organized as follows: Section 2 describes the relevant background information for this problem. Section 3 provides the experimental setup of our evaluation. Section 4 discusses our result analysis followed by conclusion and future work.

2. Background

For the English language, the coreference resolution task has seen a paradigm shift from rule based approaches [4, 5] to using word vector representation for ranking the semantic relation between entity mentions using deep learning models [6, 7, 8] with high accuracy. In the area of

Hindi language anaphora resolution, researchers have experimented with rule based technique [9] to hybrid supervised approaches [10, 11]. A generic machine learning (ML) based framework for anaphora resolution was experimented for Indo-Aryan and Dravidian languages by using the commonality between these morphologically rich languages with encouraging results [12]. Another framework for Hindi language was proposed where feature selection and ensemble learning was jointly learned using ML techniques [13].

With more linguistically diverse population using social media for opinion exchange and communication, code-mixed data processing is the focus of NLP community. NLP researchers have been exploring various aspects of code-mixed social media text. Some of the NLP problems found addressed in the literature regarding code-mixed data is discussed here briefly. Das and Gambäck [2] addressed the problem of language identification of tokens using character n-grams, dictionaries and Support Vector Machine (SVM) classification techniques in code-mixed social media data. Part Of Speech (POS) tagging using conditional random field (CRF) based classifiers is found addressed by many [3, 14, 15]. Sarcasm detection on code-mixed data was approached using supervised SVM and random forest classifier [16] and hate speech detection using sequence learning models [17]. A hybrid supervised classification approach was used for code-mixed named entity recognition (NER) [18]. Automatic normalization of word variations in code-mixed data has also been attempted using SVM based classifiers [19].

As per our knowledge, this will be the first attempt to address the problem of anaphora resolution for code-mixed social media text for Hindi language. Our result can be taken as the baseline for this problem with this dataset for further research.

3. Experimental Setup

3.1. About the Dataset

The code-mixed Hindi-English dataset for Anaphora Resolution was provided by FIRE 2020 SocAnaRes-IL[3] organizers. The data set comprised of both Hindi and English tweet documents in coNLL[4] format with antecedent and anaphora marked for training data and only text for test data in coNLL format. The training data consists of three columns with word/tokens in the first column, antecedent/anaphor markables in second and linked antecedent markable id in the third column.

The data statistics is given in Table 1. For training, a total of 810 documents of which 110 are English tweet data documents and 700 are Hindi tweet data documents. For testing, total 1205 documents are there with 654 English tweet documents and 551 Hindi tweet documents. Table 1 shows the vocabulary count and average document length in tokens for both train and test set.

3.2. Data Preprocessing

Data normalization is an important preprocessing step in problem solving. It is needed for improved model performance in deep learning techniques. In this task, we are dealing with

[3] http://78.46.86.133/SocAnaRes-IL20/

[4] http://ufal.mff.cuni.cz/conll2009-st/task-description.html

Table 1
Dataset statistics used for training and testing

Dataset	English Tweet Documents	Hindi Tweet Documents	Unique Tokens	Average Length of Documents (in tokens)
Train set	110	700	4689	62
Test set	654	551	15589	60

Twitter data which has its inherent challenges as mentioned in introduction section above. We preprocessed the data following the series of steps discussed here. The input data was in CoNLL text format with antecedent and anaphora marked columns.

- The dataset comprised of words in both Devanagari and English scripts as in example 1. The English script words converted to lowercase for processing.
- Social media text often has emoticons in between text. The dataset was cleaned of all symbols and emoticons using emoji package.
- The text Html entities such as &, " etc. were substituted with actual punctuation symbols using regular expression.
- Instances of repetitive punctuation symbols and intentional spelling variation to create some verbal effect *e.g. " sooooo goood ", " congratulations!!!!! "* is a common phenomenon in social media text. All such instances are replaced with correct spellings and one instance of the symbol using regular expression.
- The tweets were cleaned of all the url links, id_str and id of the Twitter text as it was noise to anaphora resolution.
- The text was tokenized for words and sentence for processing ahead.
- Each document was affixed with two special tokens as *<root>* and *<eos>* for marking the start and end of a document. *<root>* token at index 0 is also used as head for all tokens which were neither anaphor nor antecedent in the document.

CoNLL format data obtained after these initial cleaning steps was ready to be used for model preprocessing and training.

3.3. Model Architecture

We use an encoder-decoder with attention architecture to perform anaphora resolution for code-mixed social media data. The network takes a sequence as input and tries to output another sequence. However, we are interested in where the network attends. We train the network such that while processing anaphoras, it attends the antecedent. To this end, we have a slight modification. Our network uses one encoder and two decoders with an attention layer per each decoder. The attention layer of the first decoder learns to attend to the start index of the antecedent span, whereas the attention layer of the second decoder learns to attend to the

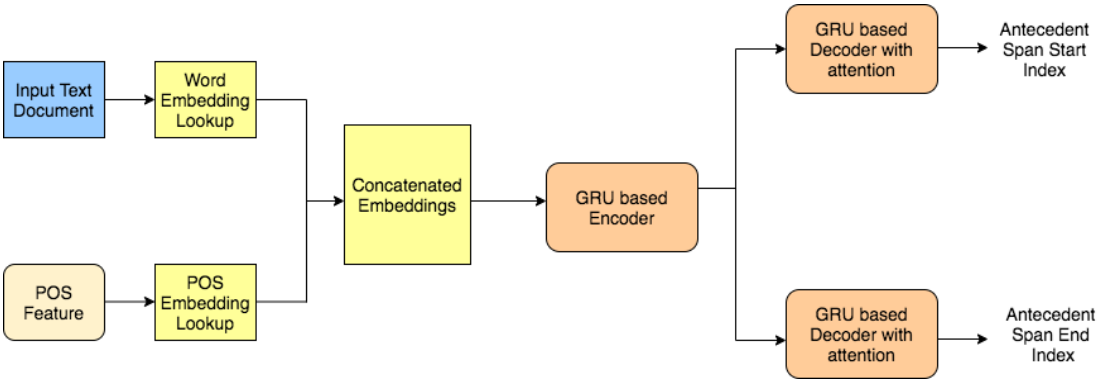

Figure 3: Model architecture block diagram

end index of the antecedent span. The loss is computed between the obtained attentions and the gold attentions (created using the anaphora-antecedent mapping from the training data)

The CoNLL data obtained after the preprocessing step was prepared for model input by processing for antecedent span information. The data study has revealed that the antecedent spans in the dataset are in the range of single token to five tokens. To include the antecedent span information for every marked anaphor, each document was added with two new columns, *antecedent span start* and *antecedent span end*. In antecedent start column, each word is mapped to *<root>* token index except the marked anaphor word, which is mapped to antecedent span start index. Similarly, for antecedent end column, each word is mapped to *<root>* token index except the anaphor word, which is mapped to the antecedent span end index. Antecedent spans of single words have the same index value at the start column and end column.

Since part of speech (POS) is an important syntactic feature in identifying the antecedent to an anaphor [12, 1], each token is annotated with POS tag using Stanfordnlp POS tagger [20] for Hindi and English language. This feature was added in CoNLL format to each token in the document.

After analyzing the train and test data it was found that only approximately 33% of the vocabulary is common between train and test set. So creating the embedding based on train set will lead to approximately 60% of unknown tokens in the test set. To address this issue of unknown tokens, both train and test set data vocabulary was used for training a custom word2Vec embedding model using gensim.

The original train set of total 810 documents was split into train/validation sets as 710/100 documents respectively for model training and tuning. The entire test set of 1205 document was kept aside for the testing purpose.

The input text data was encoded using custom word embedding created with code-mixed social media data. The POS feature was integer encoded for model training. The data thus prepared is given as input to the GRU based encoder-decoder model with attention. The model returned parallel output as antecedent start and antecedent end index values. The model architecture block diagram is shown in Figure 3.

In the code-mixed data, about 75% of documents have a token length of less than 80. So a max_len of 80 tokens is decided to standardize the variable input length. The input text

Table 2
Model result with different configurations

Model / Dataset	Validation dataset			Test dataset		
	Recall	**Precision**	**F-measure**	**Recall**	**Precision**	**F-measure**
Configure_1	0.15	0.45	0.22	0.14	0.39	0.21
Configure_2	0.15	0.52	0.23	**0.14**	**0.42**	**0.21**
Configure_3	0.16	0.53	0.24	0.13	0.41	0.20
Configure_4	**0.16**	**0.55**	**0.25**	0.13	0.42	0.20

was padded and trimmed to a max_len of 80 tokens. Similarly, the POS feature vector and antecedent start and end sequences vectors were also padded and trimmed to max_len 80 for processing. The POS feature vector and input word embedding matrix of 150 dimensions is concatenated together for input to the model. The model architecture consists of 100 GRU units for both encoder and decoder layer with multi-head attention layer. Adam optimizer [21] was used for optimizing the weights and minimizing the loss. The model was trained with four different configurations of hyperparameters by varying the loss function, learning rates, hidden units and weight coefficients for 100 epochs with early stopping feature. The model was trained with a weighted loss which penalized heavily for inaccurate predictions for non *<root>* index. The four configuration settings experimented are as mentioned:

- **Configure_1:** epochs:100, weight coefficients:1e5, loss function: Mean Square Error loss, learning rate:3e-4
- **Configure_2:** epochs:100, weight coefficients:1e4, loss function : Mean Square Error loss, learning rate:3e-3
- **Configure_3:** epochs:100, weight coefficients:1e4, loss function: Kullback-Leibler divergence loss, learning rate:3e-3
- **Configure_4:** epochs:100, weight coefficients:1e5, loss function: Kullback-Leibler divergence loss, learning rate:3e-4

All other parameters are kept the same for these configurations. The model is trained to predict the starting index and ending index of each span. We filtered the span index values of words with pronoun POS tag from the predicted index values of all tokens. The filtered span index is mapped to the token of that index. The mapped span indexes are given as the antecedent output for the test set.

4. Result Analysis

The standard evaluation metrics of Precision, Recall and F-measure was used to evaluate the predictions. Table 2 shows the results obtained from four different configurations of the model. The result shows a precision of 0.55 and 0.42 and a recall of 0.16 and 0.14 on validation and test set respectively. From high precision and low recall values, it can be inferred that the model

is predicting very few antecedent values, but most of the predicted labels are correct. The low recall value has brought the F-measure to 0.21.

On further analysis of validation set predicted output against the gold reference at a document level, it was found that from a data size of 100 documents, only 47 has marked anaphora-antecedent pair present in the actual dataset. Our best configuration model is predicting clusters in 48 documents from 100. From 48 documents marked by our model, 31 documents have correct cluster also marked among the list of clusters mapped by the model in each document. For 16 documents, our model is unable to identify any cluster making them a false negative case and in 1 case where the model has wrongly marked a cluster is a false positive scenario.

We intuit that more training data and grammatical features could address the low recall values. Since this is the first instance where an anaphora resolution is explored for code-mixed social media data in Hindi language, it can be taken as the baseline for this task.

5. Conclusion and Future work

This paper describes our approach to address the task of anaphora resolution for Hindi-English code mixed twitter data. We experimented with a GRU based encoder-decoder model with multi-headed attention to map the anaphoras with antecedents. Since anaphora resolution in itself is a challenging problem in NLP, our F-score of 0.21 for Hindi language code-mixed data is encouraging.

In future, we plan to normalize the data for embedded Hindi transliterated text and abbreviated text in specific which is a common pattern in twitter text. Due to the limited text size in twitter text, the antecedent is missing from the text. We plan to explore the inclusion of world knowledge in anaphora resolution system to resolve these missing antecedents.

References

[1] R. Sukthanker, S. Poria, E. Cambria, R. Thirunavukarasu, Anaphora and coreference resolution: A review, Information Fusion 59 (2020) 139–162.

[2] A. Das, B. Gambäck, Identifying languages at the word level in code-mixed indian social media text (2014).

[3] Y. Vyas, S. Gella, J. Sharma, K. Bali, M. Choudhury, Pos tagging of english-hindi code-mixed social media content, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 974–979.

[4] S. Lappin, H. J. Leass, An algorithm for pronominal anaphora resolution, Computational linguistics 20 (1994) 535–561.

[5] H. Lee, A. Chang, Y. Peirsman, N. Chambers, M. Surdeanu, D. Jurafsky, Deterministic coreference resolution based on entity-centric, precision-ranked rules, Computational linguistics 39 (2013) 885–916.

[6] K. Lee, L. He, M. Lewis, L. Zettlemoyer, End-to-end neural coreference resolution, arXiv preprint arXiv:1707.07045 (2017).

[7] K. Lee, L. He, L. Zettlemoyer, Higher-order coreference resolution with coarse-to-fine inference, Proceedings of NAACL (2018).

[8] M. Joshi, O. Levy, D. S. Weld, L. Zettlemoyer, Bert for coreference resolution: Baselines and analysis, Proceedings of IJCNLP (2019).
[9] S. Agarwal, M. Srivastava, P. Agarwal, R. Sanyal, Anaphora resolution in hindi documents, in: 2007 International Conference on Natural Language Processing and Knowledge Engineering, IEEE, 2007, pp. 452–458.
[10] B. Uppalapu, D. M. Sharma, Pronoun resolution for hindi, in: 7th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2009), 2009, pp. 123–134.
[11] P. Dakwale, V. Mujadia, D. M. Sharma, A hybrid approach for anaphora resolution in hindi, in: Proceedings of the Sixth International Joint Conference on Natural Language Processing, 2013, pp. 977–981.
[12] S. L. Devi, V. S. Ram, P. R. Rao, A generic anaphora resolution engine for indian languages, in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, pp. 1824–1833.
[13] U. K. Sikdar, A. Ekbal, S. Saha, A generalized framework for anaphora resolution in indian languages, Knowledge-Based Systems 109 (2016) 147–159.
[14] A. Sharma, R. Motlani, Pos tagging for code-mixed indian social media text: Systems from iiit-h for icon nlp tools contest, in: International Conference On Natural Language Processing, 2015.
[15] D. Gupta, S. Tripathi, A. Ekbal, P. Bhattacharyya, Smpost: Parts of speech tagger for code-mixed indic social media text, arXiv preprint arXiv:1702.00167 (2017).
[16] S. Swami, A. Khandelwal, V. Singh, S. S. Akhtar, M. Shrivastava, A corpus of english-hindi code-mixed tweets for sarcasm detection, arXiv preprint arXiv:1805.11869 (2018).
[17] S. Kamble, A. Joshi, Hate speech detection from code-mixed hindi-english tweets using deep learning models, arXiv preprint arXiv:1811.05145 (2018).
[18] R. Bhargava, B. Vamsi, Y. Sharma, Named entity recognition for code mixing in indian languages using hybrid approach, Facilities 23 (2016).
[19] R. Singh, N. Choudhary, M. Shrivastava, Automatic normalization of word variations in code-mixed social media text, arXiv preprint arXiv:1804.00804 (2018).
[20] P. Qi, T. Dozat, Y. Zhang, C. D. Manning, Universal dependency parsing from scratch, in: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 160–170. URL: https://nlp.stanford.edu/pubs/qi2018universal.pdf.
[21] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980 (2015).