IndIE: A Multilingual Open Information Extraction Tool For Indic Languages

Ritwik Mishra[1] , **Simranjeet Singh**[2] , **Rajiv Ratn Shah**[1] , **Ponnurangam Kumaraguru**[3] and **Pushpak Bhattacharyya** [4]
[1] IIIT, Delhi [2] NSUT, Delhi [3] IIIT, Hyderabad [3] IIT Bombay
ritwikm@iiitd.ac.in, simranjeets.ec18@nsut.ac.in
rajivratn@iiitd.ac.in, pk.guru@iiit.ac.in, pb@cse.iitb.ac.in

Abstract

Open Information Extraction (OIE) is the process of extracting informative facts from open-domain natural language text. A multilingual OIE tool, *IndIE*, has been proposed, which performs chunking, creates a Merged-phrase Dependency Tree (MDT), and generates triples using hand-crafted rules. It is observed that fine-tuned transformer-based chunker outperforms other traditional methods of chunking. A benchmark called *Hindi-BenchIE* has also been developed for automatically evaluating Hindi triples. The developed OIE tool, *IndIE*, has been automatically evaluated on the golden-triples of 112 Hindi sentences. Compared to other multilingual methods, the *IndIE* method generates more meaningful triples with 0.51 F1-score. It is observed that *IndIE* generates more fine-grained triples than other methods. It is conjectured that *IndIE* has the ability to generate meaningful triples for Urdu, Tamil, and Telugu sentences as well because the developed chunker is shown to generalize across various natural languages, and the triple generation rules are based on dependency relations that are common to the aforementioned Indic languages.

1 Introduction

India is a linguistically diverse country. Among the top 20 most spoken languages globally, six are native to India (eth, 2021; cen, 2011). Despite being spoken by billions of people, many Indic languages are considered low-resource due to the lack of annotated resources and automated systems available for them (Hirschberg and Manning, 2015). Consequently, there has also been a scarcity of tools for information extraction in Indian languages due to a lack of research work in the field (Gupta et al., 2019; Harish and Rangan, 2020).

Introduced in the mid-1960s, the concept of Information Extraction (IE) deals with extracting structured facts from unstructured text written in a natural language (WILKS, 1964). Extraction of informative facts irrespective of the text domain is called Open Information Extraction (OIE). A standard convention to represent facts is through triples <*head, relation, tail*> where *relation* denotes the link between the two entities, *head* and *tail*. For example, consider the sentence "*PM Modi to visit UAE in Jan marking 50 yrs of diplomatic ties*", one of the possible meaningful triple would be <*PM Modi*, *to visit*, *UAE*>.

The biggest strength of OIE tools is their ability to extract triples from large amounts of texts in an unsupervised manner (Gamallo et al., 2012). OIE also serves as an initial step in building or augmenting a knowledge graph out of an unstructured text (Muhammad et al., 2020; Lin et al., 2020).

A triple can be extracted in many different ways depending on the word-order constraints in the given natural language and the expected level of detail in the triples. Consider the sentence, *John sliced an apple with a knife*. Two possible ways to extract facts from this sentence are: (i) <*John, sliced, an apple with a knife*> (ii) <*John, sliced, an apple*>, and <*John, sliced, with a knife*>. Both ways represent the same fact but with different levels of detail. In the case of languages with free word order, like Hindi (Mohanan, 1994), one fact can be represented by many permutations of the elements of a triple. For example, both the following triples <राम ने, खाया, एक सेब> [<*rAm ne*, *khAya*, *ek seb*>][1] and <एक सेब, खाया, राम ने> [<*ek seb*, *khAya*, *rAm ne*>] represents the same information as the following English triple: <*Ram*, *ate*, *an apple*>. However, since the Hindi language uses postpositions (*kaarak*) instead of prepositions (Nagendra, 2019), those word permutations are prohibited that detach the postposition word from

[1]Italicized text written in square brackets represents the ITRANS transliteration, whereas the italicized text in round brackets represents the English translation of the preceding Hindi phrase/sentence

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023, pages 312–326
November 1–4, 2023.

its intended subject word because the meaning of the triple changes. For example, the following triple <एक सेब, ने खाया, राम> [<*ek seb*, *ne khAya*, *rAm*>] conveys that <*An apple*, *ate*, *Ram*>.

Our work is primarily focused on automatically extracting triples from Hindi sentences since all the authors of this work are familiar with the language. However, the proposed tool can extract triples from other low-resource Indic languages such as Tamil, Telugu, and Urdu. The main contributions of this paper are as follows:

1. We create and release an OIE benchmark dataset for Hindi sentences, *Hindi-BenchIE*, to facilitate the automatic evaluation of machine-generated triples. To our knowledge, it is the first benchmark that can handle the free-word order nature of Hindi and diverse triple extractions from different OIE systems.

2. We fine-tune a transformer model on manually annotated chunks from six natural languages (*Hindi*, *English*, *Urdu*, *Nepali*, *Gujarati*, and *Bengali*). The resulting model is able to perform chunking on languages it has not seen during the fine-tuning phase.

3. In our research, we also observed that when fine-tuning a pretrained encoder for sequence labeling tasks like chunking, taking an average of subword embeddings or taking the last subword embedding consistently outperformed the traditional way of taking the first subword embedding.

4. We propose a greedy algorithm to extract triples from Hindi text. All the resources and source code will be publicly available on https://github.com/ritwikmishra/IndIE.

2 Related Work

Earlier works have used a combination of shallow parsing and hand-crafted rules to extract meaningful entities from English language text (Mohanty et al., 2005; Etzioni et al., 2008; Christensen et al., 2011; Fader et al., 2011). Mausam et al. (2012) used hand-crafted rules and dependency parsing to develop OLLIE, which captured relations mediated by non-verbal phrases like "*is the president of*" extracted triples from English sentences. OLLIE was found to be performing at par with an SRL-based triple extractor. While most of the works dealt with extracting facts in the form of triples, KrakeN was developed to extract facts as N-ary tuples using dependency parsing (Akbik and Löser, 2012). One drawback of earlier rule-based OIE methodologies was their strictly extractive nature, i.e., triples could contain only those explicitly mentioned words in the sentence. Hence, appositive relationships[2] were not extracted by such tools (Zhan and Zhao, 2020). The method we used to extract such appositive relationships is discussed in section 3.3.

Del Corro and Gemulla (2013) developed ClausIE, which identified clauses in an English sentence and then extracted facts by classifying the identified clauses using rules. In order to identify the relations or entities, dependency parsing of sentences was a crucial step in ClausIE and many other works (White et al., 2016; Zhang et al., 2018; Gamallo et al., 2012; Gamallo and Garcia, 2015). Built as an improvement to ClausIE, the MinIE (Gashteovski et al., 2017) tool generated much more fine-grained and concise facts as compared to ClausIE. The triples generated by MinIE had dictionary-like attributes containing information about certainty, polarity, and knowledge source. Due to the availability of manually annotated data for the English, much of the recent OIE research is based on deep neural architectures where the triple extraction problem is divided into the following two sub-problems: (a) Relation extraction and (b) Argument (*head*/*tail*) extraction using features from the extracted relation (Ro et al., 2020). Span selection (using sequence labeling paradigm) is a common practice to extract relations and their corresponding arguments in such OIE methods (Zhan and Zhao, 2020).

The development of OIE tools for languages other than English is impeded by the need for annotated resources available for them. However, the field of language-independent (multilingual) OIE started in 2015 with two methods. The first method was developed by Manaal and Kumar (*M&K*) (Faruqui and Kumar, 2015), where the authors translated the source language to English using Google translate and then extracted triples using the OLLIE tool. The English triples were projected back to their source language through word alignments. It could handle as many languages as Google can translate, but machine translation has not been regarded as a sustainable solution for OIE

[2]It is a grammatical construction where two noun phrases are written adjacent to each other to convey additional information. For example: *My brother, Bob, likes ice cream*

due to translation errors (Claro et al., 2019). The second method was a rule-based triple extractor called *ArgOE* (Gamallo and Garcia, 2015). In order to generate triples, it expects dependency parse of a sentence in CoNLL format as input. However, the extracted triples contain only verb-mediated relations. *PredPatt* (White et al., 2016) was developed a year later, which also relied on a dependency parse tree and hand-crafted rules to identify predicate-argument structure in a sentence. Another work called *Multi2OIE* modeled the problem of identifying predicate-argument structure through two sequence-labeling tasks using mBERT embeddings and multi-head attention blocks (Ro et al., 2020). The first task identified all the predicates in a sentence, and the second task identified all arguments associated with each predicate. One limitation in identifying predicates with the sequence labeling paradigm is its inability to identify overlapping predicates. For example, consider the following sentence "*Nehru became the prime minister of India in 1947*". Depending on the level of detail (granularity) in triples, two predicates that can be extracted among numerous possible predicates are "*became*" and "*became the prime minister*".

Kolluru et al. (2022) introduced a novel approach to multilingual Open Information Extraction that leverages Natural Language Generation (NLG) techniques and cross-lingual projections. Their method is capable of extracting overlapping relations (predicates) and triple arguments. However, their proposed AACTrans algorithm required parallel corpora for training, and they utilized off-the-shelf translation systems in their experiments. In order to compare the performance of *IndIE*, we have taken the five methods mentioned above (*M&K, ArgOE, PredPatt, Multi2OIE, and Gen2OIE*) as our baselines since they are on similar lines as that of our work.

3 Methodology

Our method takes raw text as input and uses the Stanza library (Qi et al., 2020) (version 1.1.1) to perform sentence segmentation and dependency parsing. The primary motivation behind using the Stanza library was its ability to perform shallow parsing on multiple Indic languages. Figure 1 describes the overall procedure of generating triples. It is divided into the following three primary steps: (a) performing chunking and identifying the semantic phrases in the given sentence, (b) creating a Merged-phrases Dependency Tree using the dependency parse tree, and (c) generating triples through our hand-crafted rules. In the following subsections, we discuss the three steps in a more detailed manner.

3.1 Chunking

The process of chunking can be defined by capturing non-overlapping multi-word entities in a sentence and classifying them into different syntactic phrases (Tjong Kim Sang and Buchholz, 2000). Chunking is modeled as a sequence labeling task where one chunk tag is predicted for every token of the given sentence. Each chunk tag consists of (i) a boundary label and (ii) a chunk label. The chunk labels can be classified into different syntactic categories, like Noun-Phrases (NP), Verb-Phrases (VP), Adjective-Phrases (JJP), etc. (Bharati et al., 2006). Whereas different notations, like BIO or BIOES, can be used to represent the non-overlapping boundary labels. We use BI notation to mark boundary labels because earlier works have shown its superior precision over other notations (Singh et al., 2005; Sharma et al., 2016).

3.1.1 Dataset

We develop a multilingual chunking tool by fine-tuning a pre-trained transformer on multilingual chunk annotated data. Our chunker is fine-tuned on chunk annotated sentences from Jha (2010)[3] and Bhat et al. (2017)[4]. The former data source comprises 70K chunk-labelled sentences in English, Hindi, Bengali, Nepali, and Gujarati each, whereas the latter data source consists of 16K and 5K chunk-labelled sentences in Hindi and Urdu, respectively. The primary motivation behind using two data sources is to have a large amount of fine-tuning data. The two data sources gave us 0.37 million chunk annotated sentences in total.

3.1.2 Model

Using transformers library (Wolf et al., 2020), we fine-tune different pretrained transformer-based models for the task of chunking because earlier works have shown their superior ability to perform well on shallow parsing tasks (Tran et al., 2020; Doostmohammadi et al., 2020; Li et al., 2021). The word embeddings are obtained by taking an unweighted average of all its subword embeddings. To compare the performance of a transformer-based

[3] from http://tdil-dc.in

[4] from https://universaldependencies.org/

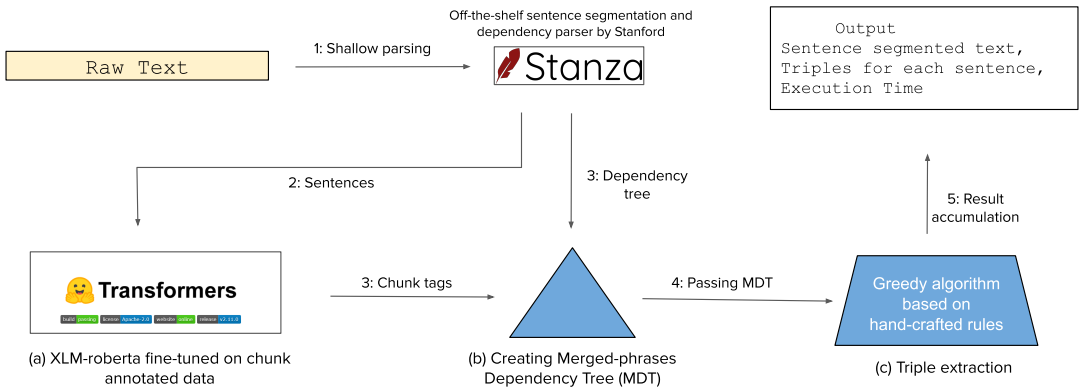

Figure 1: Overall architecture of the *IndIE* tool. The three primary steps are (a) Chunk tag prediction, (b) Creating Merged-phrases Dependency Tree (MDT), and (c) Triple generation. The three steps are run for each sentence segmented by the Stanza library (Qi et al., 2020).

chunker, we train a Conditional Random Field (CRF) model using the scikit-learn (Pedregosa et al., 2011) library. We also implemented a second-order Hidden Markov Model (HMM) with Viterbi decoding to predict the chunk tags. Both the models are the traditional methods used for chunking in Indic languages (Bharati and Mannem, 2007). Appendix B contains the implementation details for the baseline models.

A given text is parsed by the Stanza library (Qi et al., 2020), which performs sentence segmentation, POS tagging, and dependency parsing. Each segmented sentence is passed to our chunker, which predicts the chunk tags for each token. The predicted chunk tags identify the non-overlapping phrases (or multi-word expressions). A syntactically rich phrase is constructed by concatenating all the attributes of its member tokens. Each phrase is stored in a list in order of its appearance in the sentence. The list of phrases is then passed to the next step, which creates the Merged-phrases Dependency Tree.

3.2 Merged-phrases Dependency Tree (MDT)

Dependency trees have been used extensively in OIE to aid in the generation of triples from raw text (White et al., 2016; Zhang et al., 2018; Gamallo et al., 2012; Gamallo and Garcia, 2015; Del Corro and Gemulla, 2013). A traditional dependency tree is constructed at a token level, i.e., the leaves of the tree are the tokens present in the sentence, and the edges connecting them represent the dependency relation between the two tokens. A Merged-phrases Dependency Tree (*MDT*) is a coarse tree where each node contains a phrase or a multi-word expression from the sentence. An online tool by explosion.ai[5] illustrates the difference very well. One head is identified in each phrase (similar to Dobrovolskii (2021)), and the dependency relation between two heads is used as the dependency relation between the two corresponding phrases. The token-level dependency tree serves as a guiding tool to identify the dependency relationships between the identified phrases. Figure 2 illustrates the comparison between a traditional dependency tree and a generated *MDT* using an example of a Hindi sentence. It differs from a constituency tree as it does not conserve the syntactic relationships between the head and the rest of the tokens in each phrase. To the best of our knowledge, there is no publicly available tool for either constituency tree parsing or *MDT* parsing of sentences in Indic languages. Therefore, we developed a rule-based method to generate *MDT* from a traditional dependency tree.

The phenomenon of Complex Predicates (CPs) is common in Hindi, where a single action is represented by a noun-verb combination (called *conjunct verbs*) or a verb-verb combination (called *compound verbs*) (Burton-Page, 1957; Fatma, 2018). An *MDT* proves to be more useful in representing a sentence where the traditional dependency tree fails to parse CPs in languages like Hindi. For example, consider the sentence प्रारंभिक खगोलविदों का मानना था कि पृथ्वी ब्रह्मांड के केंद्र में है [*prarambhik khagolvido ka mAn-na tha ki prithvi brahm-And ke kendr me hae*] (*Early astronomers believed that Earth is in the center of the universe*), where the action of *believed* is represented by the Hindi *compound verb* मानना था

[5] https://explosion.ai/demos/displacy/

[*mAn-na tha*]. In the token-level dependency tree of this sentence, the following parent → child structure is generated: था [*tha*] (past-tense-inflection) $\xrightarrow{\text{nsubj}}$ मानना [*mAn-na*] (*to believe*), which is incorrect; the correct structure would be मानना [*mAn-na*] (*to believe*) $\xrightarrow{\text{aux}}$ था [*tha*] (past-tense-inflection). A chunker identifies *compound verbs* as a single Verb Phrase, thus generating a meaningful *MDT*. It has also been observed that, without a Multi-word Entity Recognition tool, identification of triple arguments becomes difficult by using dependency parsing alone (Gamallo et al., 2012). Gulordava and Merlo (2016) also have shown that the performance of a dependency parser degrades for natural languages having free word-order.

3.3 Triple generation

We use hand-crafted rules for capturing the *head*, *relation*, and *tail* from the *MDT* of a sentence. Similar to Mesquita et al. (2013), our hand-crafted rules are constructed by studying all the possible dependency relations in Hindi[6]. The rules are developed by carefully analyzing 80 different Hindi sentences, covering 26 out of 27 possible dependency relations in Hindi. One dependency relation that is not covered in our chosen Hindi sentences is *vocative*. Since it occurs only in 6 out of 16K dependency-annotated sentences in the data by Bhat et al. (2017), we observed that the Stanza dependency parsing tool fails to predict *vocative* relation in Hindi. In the dependency annotated data[7] of Tamil, Telugu, and Urdu, the percentage of nodes that are connected to their parents using a Hindi dependency relation are 96%, 98%, and nearly 100%, respectively. Hence, the authors are of the opinion that triple extraction rules based on Hindi dependency relations have wide coverage and could find their applicability in other Indic languages.

In the final set of rules, there are more than 100 decision-making statements (such as if-else). Therefore, we will not be explaining all the triple extraction rules here for the sake of brevity. Appendix E contains an abstracted algorithm illustrating the triple extraction procedure.

One novel property of our hand-crafted rules is their ability to capture *appositive* relationships between two entities. Earlier multilingual methods were unable to capture such *appositive* relationships. For example, in the sentence, शरमीला टैगोर के बेटे सैफ अली खान को मिला पद्म श्री पुरस्कार [*sharmila taegore ke bete saef ali khAn ko mila padm shri puraskAr*] (*Son of Sharmila Tagore, Saif Ali Khan, was awarded Padma Shri*), there exists an appositive *is-a* relationship between *Saif Ali Khan* and *Son of Sharmila Tagore*. Our system captures such appositive relationships that are expressed by *nominal modifier* (*nmod*) and *appositional modifier* (*appos*) dependency relation in the *MDT*. Our method selects the parent of these relations as <*head*>, and the child as <*tail*> of the triple. Mesquita et al. (2013) used the English auxiliary verb '*be*' to represent the <*relation*> for appositive relationships in English. We used the Hindi auxiliary verb है [*hae*] (*is/be*) to denote the <*relation*> of a triple that contains an *appositive* relationship in a Hindi sentence. The overall dataflow of the proposed architecture is illustrated in Appendix A using an example.

4 Triple Evaluation

The quality of generated triples is generally evaluated by getting them annotated by native speakers of that language. However, the procedure is time and cost intensive. Moreover, the lack of availability of Indic language annotators creates a hurdle in the manual evaluation process. On the other hand, automatic evaluation methods based on gold annotations (like CaRB (Bhardwaj et al., 2019)) do not consider the fact that there can be multiple ways to extract meaningful triples. Therefore, extending a work titled *BenchIE* (Gashteovski et al., 2021), we developed an automatic evaluation method, *Hindi-BenchIE*, based on multiple gold annotations to evaluate Hindi triples generated by any OIE tool.

Hindi-BenchIE

A natural language sentence is generally composed of one or more facts. In the original work of *BenchIE* (Gashteovski et al., 2021), multiple triples were written manually (called *golden triples*) to represent a single fact of the sentence. In our proposed benchmark, *Hindi-BenchIE*, we extend the *BenchIE* notations by introducing the following two subcategories of *golden triples*: (a) *essential-triples* and (b) *compensatory-triples*. An *essential-triple* is a triple that contains all the information needed to represent a fact. There might be some phrases in an *essential-triple* without which the

[6] https://universaldependencies.org/treebanks/hi_hdtb/index.html

[7] from https://universaldependencies.org/

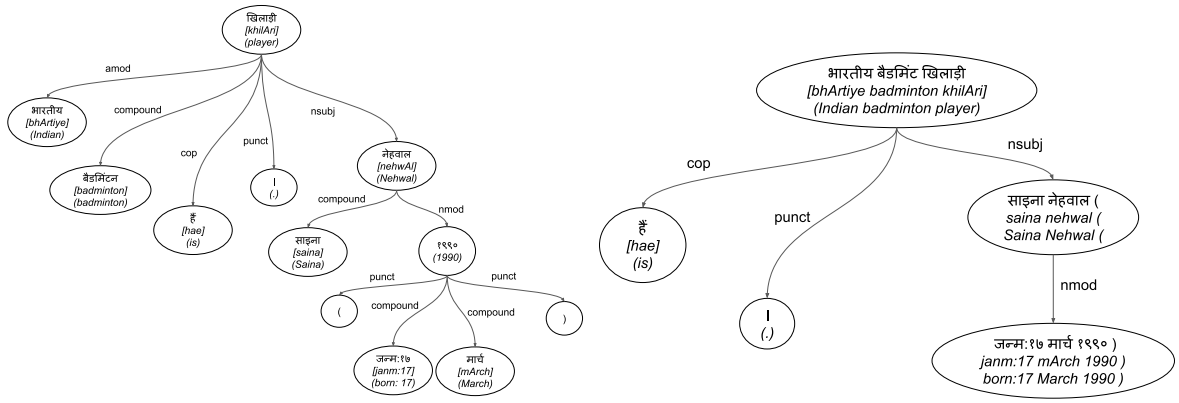

Figure 2: A comparison between a traditional dependency tree (left) and a Merged-phrases Dependency Tree (right) for the given Hindi sentence: साइना नेहवाल (जन्म:१७ मार्च १९९०) भारतीय बैडमिंटन खिलाड़ी हैं । [*saina nehwAl (janm 17 mArch 1990) bhArtiye badminton khilAri hae*] which translates to *Saina Nehwal (born:17 March 1990) is (an) Indian badminton player* . where predicted chunk labels are Noun Phrase (NP) , Verb Phrase (VP) , and Miscellaneous (BLK)

rest of the triple remains meaningful. We term such phrases as *vulnerable-phrases* in this work. However, an ideal OIE benchmark should ensure that no information is lost in the automatically generated triples. Moreover, if any information is lost, then the given OIE methodology should be penalized for it. Therefore, a *compensatory-triple* contains the information that is lost in the absence of a *vulnerable-phrase* in the generated triple. Moreover, *Hindi-BenchIE* supports the interchangeability of *head* and *tail* in a triple since Hindi is a free word-order language. These modifications facilitate annotation, as manually extracting triples for free word-order languages would otherwise require significant human effort.

In order to differentiate apposition relationships, we use an explicit keyword named '*property*' as a *relation*. In this work, a single-annotator manually extracted *golden-triples* for 112 Hindi sentences in different clusters. We release[8] the manually extracted triples for Hindi sentences since such resources are scarce in the field of multilingual OIE (Claro et al., 2019).

The number of True Positives and False Positives is calculated over all the *golden-triples* of the corresponding sentence (much like the *BenchIE*). In our work, False Negatives are calculated as the number of missing *essential-triples*, and the number of missing *compensatory-triples* that corresponds to a missing *vulnerable-phrase* (if any).

[8]https://github.com/ritwikmishra/hindi-benchie

5 Results

In order to compare our fine-tuned chunker with other traditional methods, we divided the chunk annotated data into training-set and test-set in a 50:50 ratio. We observed that the overall accuracy of our fine-tuned xlm-roberta-base (Conneau et al., 2020) chunker (91%) was superior to the other baselines of CRF (84%) and HMM (12%). The low performance on HMM is due to the sparsity in the *emission matrix* because of Out Of Vocabulary (OOV) words. Over multiple random splits, we observed that more than 80% of the test-set word bigrams were absent in the HMM training set. Due to the poor performance of HMM, we decided to use only the CRF for further comparisons. To test our chunker's multilingual nature, we curated language-specific test-sets and removed them from the training-set. This approach aligns with the principles of Leave One Language Out (LOLO) strategy, a technique documented in prior research (Ahuja et al., 2022; Srinivasan et al., 2021). Compared to CRF, the transformer-based chunker gave better accuracy on the languages it had never seen during the learning or fine-tuning phase. Table 1 compares our fine-tuned chunker and CRF chunker.

We also observed that a single linear layer and an unweighted average of subword embeddings gave the best chunking accuracy. It is important to note, however, that employing subword embedding averaging introduces a temporal overhead into the chunking process. As an alternative to the unweighted average, we observed that taking the last

Model	Hindi	English	Urdu	Nepali	Gujarati	Bengali
XLM	**78%**	**60%**	**84%**	**65%**	**56%**	**66%**
CRF	67%	56%	71%	58%	53%	53%

Table 1: A comparison of (fine-tuned) XLM chunker and CRF chunker on the languages which are removed from training-set. The numbers represent the accuracy obtained by each model when sentences from the given language are used only in the test-set. We observe that XLM chunker always perform better on unseen languages.

subword embedding is consistently better than the the conventional approach of taking the first subword embedding, a practice suggested by (Devlin et al., 2018) for Named Entity Recognition (NER) task, which is similar to chunking as both are sequence labeling tasks. For a comprehensive presentation of the results derived from our chunker ablation studies, please refer to Appendix C.

5.1 IndIE vs Others

To compare the performance of our triple extractor (*IndIE*), we used the following five baselines: (i) *M&K* (Faruqui and Kumar, 2015), (ii) *ArgOE* (Gamallo and Garcia, 2015), (iii) *PredPatt* (White et al., 2016), (iv) *Multi2OIE* (Ro et al., 2020), and (v) *Gen2OIE* (Kolluru et al., 2022) , because of their multilingual nature. The source code for *M&K* is not publicly available, but the authors have released a dataset of sentences and the corresponding triples generated by their method[9]. We randomly sampled sentences from *M&K* to create the *Hindi-BenchIE* benchmark. We used a fixed seed in order to make the random sampling reproducible.

It is essential to convey here that *PredPatt* is not designed as a triple extractor. The output generated by the method resembles an entity extractor. Appendix D shows the output of *PredPatt* on a Hindi sentence and the rules we developed to convert *PredPatt* output to triples format.

Our method, *IndIE*, performs better than other methods on *Hindi-BenchIE* golden set. Table 2 compares different OIE methods' performance. In this metric, failing to generate any triple on a given sentence penalizes the recall value of that method. In such cases, the smallest number of *essential-triples* are added to the *False Negatives* while calculating the recall. The overall results indicate that our proposed method, *IndIE*, extracted more meaningful triples from Hindi sentences than other multilingual OIE tools.

[9] https://www.kaggle.com/shankkumar/multilingualopenrelations15

6 Discussion

Motivated by qualitative observations, Table 3 presents quantitative insights about the generated triples from various methods. We observed that methods such as *ArgOE* and *PredPatt* generated more coarse triples than other methods. Coarse triples have a high sentence coverage percentage. They identify the root action in the sentence, and the remaining text is placed in the argument of triple. For example, for the following sentence 007 के गुप्त नाम से प्रसिद्ध यह एजेंट फ़्लेमिंग की बारह पुस्तकों व दो लघुकथाओं में मौजूद है। [*007 ke nAm se prasidha yeh Ejant phleming ki bArah pustakon va do laghukathaon me maaujUd hae*] (*Renowned by the name of 007, this agent appears in twelve books and two short stories by Fleming.*). The triple extracted by *ArgOE* is as follows: <007 के गुप्त नाम से प्रसिद्ध यह एजेंट, है, फ़्लेमिंग की बारह पुस्तकों व दो लघुकथाओं में मौजूद> [*<007 ke nAm se prasidha yeh Ejant, hae, phleming ki bArah pustakon va do laghukathaon me maaujUd>*] (*<Renowned by the name of 007 this agent, is, present in twelve books and two short stories by Fleming>*). Fine-grained triples are essential for downstream tasks, such as creating a knowledge-base from raw text (Zhang et al., 2021), whereas coarse triples could result in overspecific relations or entities.

The yield of triples by *Gen2OIE* method is better than other methods. However, since the source code of *M&K* is unavailable, we cannot determine the number of sentences for which the method returns no triples. Since *M&K* and *Gen2OIE* method generates triples in English and then uses word alignments to obtain Hindi triples, they often generate non-meaningful Hindi triples because the incorrect word alignments separate the postpositional word (*kaarak*) from its preceding word. As a result, more triples are generated with misplaced *kaarak*. For example, for the given sentence जब कोई मतैक्य नहीं हुआ तो विक्रम ने एक हल सोचा। [*jab koi mataekya nahin hua to vikram ne ek hal sochA*] (*When there was no consensus, Vikram thought of a solution.*), the *Gen2OIE* method generated a triple as <विक्रम, सोचा, ने एक हल> [*<vikram, sochA, ne ek hal>*] (Ungrammatical). Similar to *Gen2OIE*, the *IndIE* can generate overlapped arguments in the extracted triples. For instance, the two triples generated for the following sentence मैं शब्द की आत्मा

	ArgOE	*M&K*	*Multi2OIE*	*PredPatt*	*Gen2OIE*	*IndIE*
Precision	0.17	0.07	0.005	0.22	0.23	**0.49**
Recall	0.04	0.08	0.01	0.05	0.35	**0.53**
F1-score	0.07	0.08	0.008	0.09	0.28	**0.51**

Table 2: Performance of different OIE methods on *Hindi-BenchIE* golden set. It is observed that *IndIE* outperforms other methods on the *Hindi-BenchIE* golden set.

	ArgOE	*M&K*	*Multi2OIE*	*PredPatt*	*Gen2OIE*	*IndIE*
# Triples	51	199	59	48	**278**	277
# Sentences with no triples	69	NA	68	66	**0**	2
Avg. Tokens in a Triple	12	10	**7**	12	10	**7**
Avg. Sentence Coverage of a Triple	73%	64%	49%	76%	66%	**46%**
Triples with misplaced *kaarak*	1.9%	44%	20%	39%	16%	**0.7%**

Table 3: Triple statistics of different OIE methods on *Hindi-BenchIE* golden set of 112 sentences. Non-unique tokens are considered while counting the number of tokens in a triple. Sentence coverage is calculated by $1 - |unique(sent) - unique(triple)|/|unique(sent)|$. It can be observed that *IndIE* triples have the least sentence coverage and *kaarak* errors. Hence, *IndIE* generates more fine-grained triples than other methods.

समझकर ही इस श्रेष्ठ तत्त्व की उपासना करता हूं। [*main shabd ki Atma samajhkar hee is shreshth tatv ki upAsna karta hun*] (*I worship this supreme element after understanding the soul of the word.*) are as follows <मैं,उपासना करता हूं,इस श्रेष्ठ तत्त्व की> [<*main*, *upAsna karta hun*, *is shrestha tatv ki*>] (<*I*, *worship*, *this supreme element*>) and <इस श्रेष्ठ तत्त्व की उपासना करता हूं,समझकर ही,आत्मा> [<*is shreshtha tatv ki upAsna karta hoon*, *samajhkar hee*, *Atma*>] (<*worship this supreme element*, *after understanding*, *soul*>).

In our experiments, the zero-shot *Multi2OIE* method performs poorly on every metric, which is expected since neural methods are known to generate incorrect facts as compared to rule-based methods (Gashteovski et al., 2021). Therefore, a promising direction is to train a neural OIE method based on the output of a rule-based OIE tool for a given language.

6.1 Limitations

The utilization of hand-crafted rules in the triple extraction process imposes constraints on the scalability and versatility of the IndIE pipeline. Furthermore, while we provide a rationale supporting the potential application of the IndIE tool to other Indic languages, we encountered challenges in creating a benchmark akin to Hindi-BenchIE due to a scarcity of annotators for these languages. Consequently, the performance of IndIE on other Indic languages remains a matter of conjecture. The number of sentences in our automatic evaluation benchmark, *Hindi-BenchIE*, is much smaller than the original work of *BenchIE*. Since manually generating triples requires more effort than triple annotation, the single-annotator of *Hindi-BenchIE* was able to generate more than 500 triples for 112 Hindi sentences only. Hence, we believe that the benchmark can be further refined with the efforts of the Indic-nlp community. We also acknowledge the fact that the multilingual nature of *IndIE* is limited to the intersection between the set of languages on which xlm-roberta-base has been pretrained and the set of languages supported by the Stanza library.

7 Conclusion

The low resource nature of Indic languages has been an impediment in the development of their NLP tools. In this work, we developed an Open Information Extraction (OIE) tool, *IndIE*, that generates triples from unstructured Hindi sentences. It first predicts the chunk tags for the given sentence and then creates a Merged-Phrase Dependency Tree (MDT) to generate the triples using the hand-crafted rules. We used a multilingual pretrained transformer model and fine-tuned it with chunk annotated sentences from English and five Indic languages. It was observed that, in sequence labeling tasks (such as chunking), taking the average of subword token embeddings is more valuable than other paradigms. We created *Hindi-BenchIE*, a benchmark for automatically evaluating Hindi triples based on a set of 112 Hindi sentences to compare the performance of various multilingual

OIE tools. It was observed that the *IndIE* generates more informative and fine-grained triples than other baselines.

7.1 Future Work

We plan to explore different methods to merge the fine-grained triples to make them more informative. Further linguistic efforts are needed to analyze and capture the appositive relationships in Agglutinative Indic languages such as Tamil and Telugu. Expanding the golden triples in *Hindi-BenchIE* and developing similar benchmarks for other Indic languages is also an important direction to keep the field of OIE in Indic languages alive. Co-reference resolution is an important area to explore to generate more meaningful triples with resolved pronominal references. Moreover, the viability of OIE-based approaches needs to be explored where the length of input text sequence exceeds the capability of transformer-based models, like open-domain Question-Answering and document-level textual similarity.

Acknowledgements

Ritwik Mishra wishes to extend his appreciation to the University Grant Commission (UGC) of India for their partial support through the UGC Junior Research Fellowship (JRF) program. He would also like to express his gratitude to ACM-India for his selection as a recipient of the Anveshan-Setu fellowship and for assigning Prof. Pushpak Bhattacharya as his mentor. Rajiv Ratn Shah is partly supported by the Infosys Center for AI, the Center of Design and New Media, and the Center of Excellence in Healthcare at IIIT Delhi.

References

2011. Census of india.

2021. What are the top 200 most spoken languages?

Kabir Ahuja, Antonios Anastasopoulos, Barun Patra, Graham Neubig, Monojit Choudhury, Sandipan Dandapat, Sunayana Sitaram, and Vishrav Chaudhary. 2022. The sumeval 2022 shared task on performance prediction of multilingual pre-trained language models. In Proceedings of the First Workshop on Scaling Up Multilingual Evaluation, pages 1–7.

Alan Akbik and Alexander Löser. 2012. Kraken: N-ary facts in open information extraction. In Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX), pages 52–56.

Akshar Bharati and Prashanth R Mannem. 2007. Introduction to shallow parsing contest on south asian languages. In Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL), pages 1–8. Citeseer.

Akshar Bharati, Rajeev Sangal, Dipti Misra Sharma, and Lakshmi Bai. 2006. Anncorra: Annotating corpora guidelines for pos and chunk annotation for indian languages. LTRC-TR31, pages 1–38.

Sangnie Bhardwaj, Samarth Aggarwal, and Mausam Mausam. 2019. CaRB: A crowdsourced benchmark for open IE. In Proceedings of the 2019 Conference on EMNLP-IJCNLP, pages 6262–6267, Hong Kong, China. Association for Computational Linguistics.

Riyaz Ahmad Bhat, Rajesh Bhatt, Annahita Farudi, Prescott Klassen, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, Ashwini Vaidya, Sri Ramagurumurthy Vishnu, et al. 2017. The hindi/urdu treebank project. In Handbook of linguistic annotation, pages 659–697. Springer.

John Burton-Page. 1957. Compound and conjunct verbs in hindi1. Bulletin of the School of Oriental and African Studies, 19(3):469–478.

Janara Christensen, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In Proceedings of the sixth international conference on Knowledge capture, pages 113–120.

Daniela Barreiro Claro, Marlo Souza, Clarissa Castellã Xavier, and Leandro Oliveira. 2019. Multilingual open information extraction: Challenges and opportunities. Information, 10(7):228.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In ACL.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In Proceedings of the 22nd international conference on World Wide Web, pages 355–366.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. CoRR, abs/1810.04805.

Vladimir Dobrovolskii. 2021. Word-level coreference resolution. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7670–7675.

Ehsan Doostmohammadi, Minoo Nassajian, and Adel Rahimi. 2020. Persian ezafe recognition using transformers and its role in part-of-speech tagging. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 961–971.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. Communications of the ACM, 51(12):68–74.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In Proceedings of the 2011 conference on empirical methods in natural language processing, pages 1535–1545.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In North American Chapter of the ACL: Human Language Technologies, pages 1351–1356.

Shamim Fatma. 2018. Conjunct verbs in hindi. Trends in Hindi Linguistics, pages 217–244.

Pablo Gamallo and Marcos Garcia. 2015. Multilingual open information extraction. In Portuguese Conference on Artificial Intelligence, pages 711–722. Springer.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In Proceedings of the joint workshop on unsupervised and semi-supervised learning in NLP, pages 10–18.

Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. 2017. Minie: minimizing facts in open information extraction. Association for Computational Linguistics.

Kiril Gashteovski, Mingying Yu, Bhushan Kotnis, Carolin Lawrence, Goran Glavas, and Mathias Niepert. 2021. Benchie: Open information extraction evaluation based on facts, not tokens. arXiv preprint arXiv:2109.06850.

Kristina Gulordava and Paola Merlo. 2016. Multilingual dependency parsing evaluation: a large-scale analysis of word order properties using artificial data. Transactions of the Association for Computational Linguistics, 4:343–356.

Vaishali Gupta, Nisheeth Joshi, and Iti Mathur. 2019. Advanced machine learning techniques in natural language processing for indian languages. In Smart Techniques for a Smarter Planet, pages 117–144. Springer.

BS Harish and R Kasturi Rangan. 2020. A comprehensive survey on indian regional language processing. SN Applied Sciences, 2(7):1–16.

Julia Hirschberg and Christopher D Manning. 2015. Advances in natural language processing. Science, 349(6245):261–266.

Kushal Jain, Adwait Deshpande, Kumar Shridhar, Felix Laumann, and Ayushman Dash. 2020. Indic-transformers: An analysis of transformer language models for indian languages. arXiv preprint arXiv:2011.02323.

Girish Nath Jha. 2010. The TDIL program and the Indian langauge corpora intitiative (ILCI). In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta. European Language Resources Association (ELRA).

Keshav Kolluru, Muqeeth Mohammed, Shubham Mittal, Soumen Chakrabarti, et al. 2022. Alignment-augmented consistent translation for multilingual open information extraction. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2502–2517.

Hongwei Li, Hongyan Mao, and Jingzi Wang. 2021. Part-of-speech tagging with rule-based data preprocessing and transformer. Electronics, 11(1):56.

Xueling Lin, Haoyang Li, Hao Xin, Zijian Li, and Lei Chen. 2020. Kbpearl: a knowledge base population system supported by joint entity and relation linking. Proceedings of the VLDB Endowment, 13(7):1035–1049.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 447–457.

Tara Mohanan. 1994. Case ocp: A constraint on word order in hindi. Theoretical perspectives on word order in South Asian languages, 185:216.

Rajat Mohanty, Anupama Dutta, and Pushpak Bhattacharyya. 2005. Semantically relatable sets: building blocks for representing semantics. In MT Summit, volume 5. Citeseer.

Iqra Muhammad, Anna Kearney, Carrol Gamble, Frans Coenen, and Paula Williamson. 2020. Open information extraction for knowledge graph construction. In International Conference on Database and Expert Systems Applications, pages 103–113. Springer.

Jaya S Nagendra. 2019. Basic grammar, hindi. In A Brief History of Languages, volume 1, page 190. Atlantic.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12), pages 2089–2096.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations.

Youngbin Ro, Yukyung Lee, and Pilsung Kang. 2020. Multi^{2}oie: Multilingual open information extraction based on multi-head attention with bert. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 1107–1117.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. Transactions of the Association for Computational Linguistics, 8:842–866.

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Shrivastava, Radhika Mamidi, and Dipti Misra 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.

Akshay Singh, Sushma Bendre, and Rajeev Sangal. 2005. Hmm based chunker for hindi. In Companion Volume to the Proceedings of Conference including Posters/Demos and tutorial abstracts.

Anirudh Srinivasan, Sunayana Sitaram, Tanuja Ganu, Sandipan Dandapat, Kalika Bali, and Monojit Choudhury. 2021. Predicting the performance of multilingual nlp models. arXiv preprint arXiv:2110.08875.

EF Tjong Kim Sang and S Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. ACL.

Thi Oanh Tran, Phuong Le Hong, et al. 2020. Improving sequence tagging for vietnamese text using transformer-based neural models. In Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation, pages 13–20.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1713–1723.

YORICK WILKS. 1964. Text searching with templates. Cambridge Language Research Unit Memo, ML, (165).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Junlang Zhan and Hai Zhao. 2020. Span model for open information extraction on accurate corpus. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 9523–9530.

Sheng Zhang, Xutai Ma, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2018. Cross-lingual decompositional semantic parsing. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1664–1675.

Zixuan Zhang, Nikolaus Parulian, Heng Ji, Ahmed Elsayed, Skatje Myers, and Martha Palmer. 2021. Fine-grained information extraction from biomedical literature based on knowledge-enriched abstract meaning representation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 6261–6270.

A Illustration

The overall dataflow of the proposed architecture is illustrated using the following raw (multi-sentence) text in Hindi: शर्मीला टैगोर के बेटे सैफ अली खान को 2010 में पद्म श्री पुरस्कार मिला। वह एक भारीतय अभिनेता है। [*sharmila taegor ke bete saef ali khAn ko 2010 me padm shri puraskAr mila. veh ek bhArtiye abhinetA hae*] *(Son of Sharmila Tagore, Saif Ali Khan, was awarded with Padma Shri award in 2010. He is an Indian actor).* The raw text is fed to the Stanza library for sentence segmentation and dependency parsing of each sentence. In stage (a) of the architecture, as shown in Figure 1, segmented sentences are passed to the chunking model to predict the chunk tags for each word of the given sentence. Predicted chunked phrases from the stage (a) are as follows:

Sentence 1 - {शर्मीला टैगोर के [*sharmila taegore ke*] *(Sharmila Tagore's)*}_NP , {बेटे [*bete*] *(son)*}_NP , {सैफ अली खान को [*saef ali khAn ko*] *(to Saif Ali Khan)*}_NP , {2010 में [*2010 me*] *(in 2010)*}_NP , {पद्म श्री

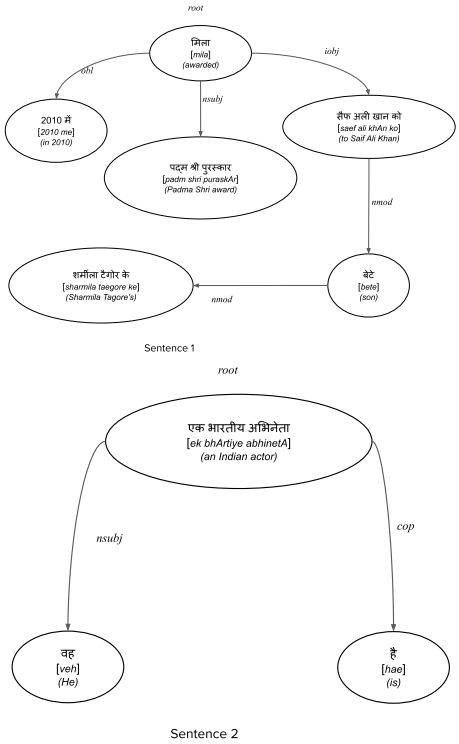

Figure 3: The generated *MDTs* after sentence segmentation, chunking, and dependency parsing of the following raw text: शर्मीला टैगोर के बेटे सैफ अली खान को 2010 में पद्म श्री पुरस्कार मिला। वह एक भारीतय अभिनेता है। [*sharmila taegor ke bete saef ali khAn ko 2010 me padm shri puraskAr mila. veh ek bhArtiye abhinetA hae*] *(Son of Sharmila Tagore, Saif Ali Khan, was awarded with Padma Shri award in 2010. He is an Indian actor).*

पुरस्कार [*padm shri puraskAr*] *(Padma Shri award)*}_NP , {मिला [*mila*] *(awarded)*}_VGF

Sentence 2 - {वह [*veh*] *(He)*}_NP , {एक भारीतय अभिनेता [*ek bhArtiye abhinetA*] *(an Indian actor)*}_NP , {है [*hae*] *(is)*}_VGF

The chunked phrases and dependency tree for each sentence are passed to the stage (b) of the architecture to construct *MDT* for each sentence. Figure 3 illustrates the *MDTs* generated at the output of stage (b). For each sentence, triples are generated using its corresponding *MDT* and our hand-crafted rules. All the triples extracted by the *IndIE* tool for the aforementioned raw text are shown in Table 4. The output of stage (c) consists of the following three types: (i) a list of segmented sentences, (ii) extracted triples, and (iii) execution time for each sentence.

	<*head*>	<*relation*>	<*tail*>
Sentence 1	पद्म श्री पुरस्कार [*padm shri puraskAr*] *(Padma Shri award)*	मिला [*mila*] *(awarded)*	सैफ अली खान को [*saef ali khAn ko*] *(to Saif Ali Khan)*
	2010 में [*2010 me*] *(in 2010)*	मिला [*mila*] *(awarded)*	पद्म श्री पुरस्कार [*padm shri puraskAr*] *(Padma Shri award)*
	सैफ अली खान को [*saef ali khAn ko*] *(to Saif Ali Khan)*	है [*hae*] *(is)*	बेटे [*bete*] *(son)*
	बेटे [*bete*] *(son)*	है [*hae*] *(is)*	शर्मीला टैगोर के [*sharmila taegore ke*] *(Sharmila Tagore's)*
Sentence 2	वह [*veh*] *(He)*	है [*hae*] *(is)*	एक भारीतय अभिनेता [*ek bhArtiye abhinetA*] *(an Indian actor)*

Table 4: Triples extracted through hand-crafted rules of the proposed IndIE tool for the following raw text in Hindi: शर्मीला टैगोर के बेटे सैफ अली खान को 2010 में पद्म श्री पुरस्कार मिला। वह एक भारीतय अभिनेता है। [*sharmila taegor ke bete saef ali khAn ko 2020 me padm shri puraskAr mila. veh ek bhArtiye abhinetA hae*] *(Son of Sharmila Tagore, Saif Ali Khan, was awarded with Padma Shri award in 2010. He is an Indian actor).*

B Chunking Baselines

We used the scikit-learn scikit-learn python library to implement[10] the CRF model. The following features were used for each word of the sentence: (a) bias ← 1.0, (b) word text, (c) POS tag of the word, (d) POS tags of preceding two words, and (e) POS tags of succeeding two words. The values of L1 and L2 regularization were obtained through grid search. We used the word text and its POS tag as features for the HMM model.

Our fine-tuned chunker is an end-to-end model for chunking because it takes input in raw text. However, CRF and HMM model expects already POS tagged sentences. The chunk annotated sentences from the Bhat et al. (2017) are POS tagged in upos format (Petrov et al., 2012), whereas sentences from the Jha (2010) are POS tagged with a scheme called AnnCorra (Bharati et al., 2006). It is an extension to the Penn tagset and tailored for Indian languages. In the absence of a publicly available POS tagger for AnnCorra, we created a mapping from AnnCorra to upos tagset for standardizing the POS tags in the entire dataset. We passed all the Hindi and English sentences from the (**?**) dataset through Stanza library, generating POS tags in upos format. Therefore, we create a mapping from AnnCorra (Penn tagset) to upos tagset which helped us in standardizing the POS

[10] https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html

tag format across all sentences.

C Chunking Ablation

We experimented with three approaches to handle sub-word token embeddings and observed that averaging sub-word token embeddings gave better accuracy as compared to the first sub-word token embedding or last sub-word token embedding, as shown in Table 5. Since averaging the sub-word token embeddings requires additional processing, its fine-tuning time (6 hours/epoch) and inference time (29 milliseconds/sentence) is more than fine-tuning time (45 minutes/epoch) and inference time (17 milliseconds/sentence) for the other two approaches. Hence, if accuracy is preferred over inference time, then we recommend taking an average of sub-word token embeddings for a sequence labeling task; otherwise simply taking the last sub-word token embedding gives better performance with equal inference time than the traditional technique of taking the first sub-word token embedding. In our experiments, we used the embeddings from `last_hidden_state` of the model. However, since some methods in the literature suggest that early layers of a transformer are responsible for learning shallow features of the text (Rogers et al., 2020), we experimented by taking average embeddings of the first two hidden layers of the model as well. It turns out that using embeddings from early layers actually decreased the accuracy (86%) for the chunking task. Confirming the findings of Jain et al. (Jain et al., 2020), we observed that xlm-roberta-base (Conneau et al., 2020) gave the best accuracy (92%) over other pretrained models.

D PredPatt

Figure 4 shows the output of *PredPatt* on a Hindi sentence. Table 6 contains the rules we developed to convert *PredPatt* output to triples format.

E Algorithm

label: sent_91
sentence: अभ्रक आदि की खानों में मोमबत्तियाँ भी प्रयुक्त होती है .
[*abhrak Adi ki khAno me mombattiya bhi prayukt hoti hae*]
(*In the mines of Mica candles are also used*)

ppatt:
अभ्रक की खानों में ?a प्रयुक्त होती है
?a: मोमबत्तियाँ
[*abhrak ki khAno me ?a prayukt hoti hae*
?a: mombattiya]
(*In the mines of Mica ?a are used*
?a: candles)

Figure 4: Output of *PredPatt* on a Hindi sentence अभ्रक आदि की खानों में मोमबत्तियाँ भी प्रयुक्त होती है [*abhrak Adi ki khAno me mombattiya bhi prayukt hoti hae*] (*In the mines of Mica candles are also used*). In the given sentence, *candles* is the *Entity1*, and it is represented with '?a' notation.

Triple generator algorithm from Merged-phrase Dependency Tree (MDT)

function EXTRACT(MDT, t, Q)
  if '*cop*' $\in t.children.dep\_rel$ **then**
    if $|t.children| \leq 2$ **then**
      $Head, Tail \leftarrow$ FIND_HEAD(MDT, t) , t
      $Rel \leftarrow x$ where ($x \in t.children \wedge x.dep\_rel =$ '*cop*')
    else
      $Head, Tail \leftarrow$ FIND_HEAD(MDT, t) , FIND_TAIL(MDT, t)
      $Rel \leftarrow t + x$ where ($x \in t.children \wedge x.dep\_rel =$ '*cop*')
    end if
  else if '*advcl*' $== t.dep\_rel$ **then**
    $Head \leftarrow q.Tail + q.Rel$ where ($q \in Q \wedge t.parent \in q$)
    $Rel, Tail \leftarrow t$, FIND_TAIL(MDT, t)
  else if '*acl*' $== t.dep\_rel$ **then**
    $Head \leftarrow t.closest\_phrase(q.Tail,\ q.Head)$ where ($q \in Q \wedge t.parent \in q$)
    $Rel, Tail \leftarrow t$, FIND_TAIL(MDT, t)
  else if '*conj*' $== t.dep\_rel$ **then**
    if $\exists q \in Q$ such that $q.Head == t.parent$ **then**
      $Head, Rel, Tail \leftarrow t$, $q.Rel$, $q.Tail$
    else if $\exists q \in Q$ such that $q.Tail == t.parent$ **then**
      $Head, Rel, Tail \leftarrow q.Head$, $q.Rel$, t
    end if
  else
    if $t.is\_clausal() == True$ **then**
      $Head, Rel, Tail \leftarrow t.pronoun$, $t.verb$, $t - (t.pronoun \cap t.verb)$
    else
      if $t.is\_a\_relationship() == True$ **then** //appositive relationship
        $Head, Rel, Tail \leftarrow t$, FIND_TAIL(MDT, t) , $t.is\_a\_label$
      else
        $Head, Rel, Tail \leftarrow$ FIND_HEAD(MDT, t) , t , FIND_TAIL(MDT, t)
      end if
    end if
  end if
  if $Head, Rel, Tail$ **then**
    $Q.add([Head,\ Rel,\ Tail])$
  end if
  if $t.contain\_args() == True$ **then**
    $Q \leftarrow$ EXTRACT(MDT, t)
  else
    for each $t_c \in t.children$ **do**
      $Q \leftarrow$ EXTRACT(MDT, t_c)
    end for
  end if
  return Q
end function

$triples \leftarrow$ EXTRACT($MDT, t_{root}, \{\}$)

Classification Layers	First sub-word token embedding	Last sub-word token embedding	Average embedding of all sub-word tokens
1	82±10 (50±20)	89±0.5 (62±1.0)	**91±0.0 (65±0.5)**
2	86±1.8 (51±6.2)	89±0.5 (54±7.4)	**90±0.5 (54±4.5)**
3	79±14 (43±13)	82±11 (41±12)	**90±0.5 (48±2.2)**

Table 5: A comparison of three approaches for solving the sub-word token embeddings for chunking task. Four different random seeds were used to calculate the mean and standard deviation for the given samples. All the experiments were run on the combined data of Jha et al. (Jha, 2010) and Bhat et al.(Bhat et al., 2017). The numbers written outside round brackets represent the accuracy, whereas numbers inside round brackets represent the macro average.

Rule No.	**Sentence Structure**	**Extracted Triple**
1	^ phrase1 Entity1 phrase2 $	<phrase1 , phrase2 , Entity1>
2	^ Entity1 phrase1 Entity2 phrase2 $ ^ Entity1 Entity2 phrase1 $ ^ phrase1 Entity1 Entity2 $ ^ Entity1 phrase1 Entity2 $	<Entity1 , phrase1 , Entity2>
3	^ phrase1 Entity1 Entity2 phrase2 $ ^ phrase1 Entity1 phrase2 Entity2 $ ^ phrase1 Entity1 phrase2 Entity2 phrase3 $	<Entity1 , phrase2 , Entity2>
4	Any other sentence structure	Discard

Table 6: The rules we used to extract triples from *PredPatt* output. Considering the sentence given in Figure 4, Rule number 1 is applied to the sentence. The symbol ^ indicates the start of the sentence, and the symbol $ indicates the end of the sentence.