CompactIE: Compact Facts in Open Information Extraction

Farima Fatahi Bayat
University of Michigan
farimaf@umich.edu

Nikita Bhutani
Megagon Labs
nikita@megagon.ai

H. V. Jagadish
University of Michigan
jag@umich.edu

Abstract

A major drawback of modern neural OpenIE systems and benchmarks is that they prioritize high coverage of information in extractions over compactness of their constituents. This severely limits the usefulness of OpenIE extractions in many downstream tasks. The utility of extractions can be improved if extractions are compact and share constituents. To this end, we study the problem of identifying compact extractions with neural-based methods. We propose CompactIE, an OpenIE system that uses a novel pipelined approach to produce compact extractions with overlapping constituents. It first detects constituents of the extractions and then links them to build extractions. We train our system on compact extractions obtained by processing existing benchmarks. Our experiments on CaRB and Wire57 datasets indicate that CompactIE finds 1.5x-2x more compact extractions than previous systems, with high precision, establishing a new state-of-the-art performance in OpenIE.

1 Introduction

A popular domain-agnostic paradigm to structure the raw text is open information extraction (OpenIE) (Banko et al., 2007). Not relying on any predefined schema, OpenIE systems typically extract information as (subject; relation; object) triples. The extracted information is then used in several downstream applications, including answering questions (Khot et al., 2017), summarizing documents (Hao et al., 2018; Ji et al., 2013), and populating knowledge bases (Fan et al., 2019).

Despite much progress, state-of-the-art neural OpenIE systems focus on covering more information from the input sentence often at the cost of utility and compactness of the extracted triples. The extracted triples have long, over-specific *constituents* (i.e. the relation and its arguments). Figure 1 illustrates such example triples produced by a popular

Beth is the second child of Henry, born in wedlock.	
IMoJIE Extractions	E1: (Beth ; is ; the second child of Henry **born** in wedlock)
Compact Extractions	E1: (Beth ; is ; the second child of Henry) E2: (the second child of Henry ; born ; in wedlock)
The rest of the group reach a small shop , where the crocodile breaks through a wall and devours Annabelle.	
IMoJIE Extractions	E1: (The rest of the group ; reach ; a small shop , where the crocodile **breaks** through a wall and **devours** Annabelle) E2: (the crocodile ; devours ; Annabelle a small shop)
Compact Extractions	E1: (The rest of the group ; reach ; a small shop) E2: (crocodile ; breaks ; through a wall) E3: (crocodile ; devours ; Annabelle)

Figure 1: Example sentences with non-compact triples from IMoJIE vs. compact triples from our benchmark. Compact triples can share constituents. Constituents for subjects, relations and objects are indicated in blue, green and orange, respectively.

OpenIE system, IMoJIE (Kolluru et al., 2020b). As shown, the knowledge that *the second child of Henry* was born *in wedlock* is embedded in a long argument. This can be problematic for downstream applications, especially knowledge base population (Gashteovski et al., 2020; Stanovsky et al., 2015) that derive power from merging multiple pieces of information extracted about the same entity. In contrast, the compact extractions are more pliable for tasks such as identifying similar facts and merging facts that share constituents. For example, compact extractions in Figure 1 can be merged to derive that *Beth* is born *in wedlock*.

Although some prior work (Corro and Gemulla, 2013; Gashteovski et al., 2017; Bhutani et al., 2016) has explored the compactness of OpenIE triples, these systems are rule-based and have been superseded by end-to-end neural OpenIE systems. In this work, we study the problem of identifying compact extractions with neural-based methods. Inspired by (Corro and Gemulla, 2013), we define an extracted triple to be *compact* if it does not contain information that can be independently represented in another triple. To further improve the suitability of

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 900 - 910

compact triples for knowledge base population, we require compact triples extracted from a sentence to have overlapping constituents.

Existing neural systems adopt a sequence labeling (Kolluru et al., 2020a; Wang et al., 2021; Ro et al., 2020) or a sequence generation (Kolluru et al., 2020b) approach to identify triples and their constituents, typically all at once, or through a pipeline that first identifies the relations and then their corresponding arguments. None of these methods guarantee that the extracted triples will be compact and share constituents.

We propose a novel pipeline system for finding compact triples that share their constituents. We call our OpenIE system, CompactIE. To encourage the constituents to be shared across triples, CompactIE first extracts the constituents using a *Constituent Extraction* model and then links them using a *Constituent Linking* model to obtain triples.

We adapt a table filling method (Wang et al., 2021) with a new schema for identifying both constituent boundaries and their roles (i.e., subject or object). This allows the constituent extraction model to capture interactions among constituents and minimize ambiguities in boundary detection. For the task of constituent linking, we train a model that builds on contextual representations specific to a given pair of constituents and predicts their relation type. Such a two-step approach enables us to optimize the models for each sub-task with different objectives and also promote the constituent reuse across triples.

Existing neural OpenIE systems are trained on benchmarks that combine extractions from multiple OpenIE systems . However, no such large-scale benchmark exists for compact triples. We develop a new benchmark using a subset of sentences in the OpenIE2016 benchmark (Mausam, 2016). Specifically, we develop a data processing algorithm that targets extraction from individual clauses in a sentence. Given an input sentence, it identifies clauses and then uses OpenIE systems such as IMoJIE over the clauses to find compact triples. We train CompactIE on the new benchmark.

Our experiments on a fine-grained benchmark, Wire57, show that CompactIE outperforms existing non-neural and neural systems by 5.8 F1 pts and 7.1 F1 pts, respectively. Manual evaluation over a coarse-grained benchmark, CaRB, indicates that CompactIE produces 1.5x-2x more compact extractions than existing systems with comparable precision, establishing a new state-of-the-art for the OpenIE task[1].

2 Background and Preliminaries

Given a sentence $s = w_1 w_2 ... w_n$, an OpenIE system generates triples of the form *(subject; relation; object)*, where *subject*, *relation* and *object* are the constituents of a triple.

2.1 Extracting Compact Triples

A recent study (Gashteovski et al., 2020) shows that triples from modern neural OpenIE systems are difficult to align to knowledge bases such as DBpedia. Less than 77% of triples from neural OpenIE systems had the same arguments as DBpedia facts. In contrast, the corresponding alignment ratio for some of the non-neural OpenIE systems was as high as 98%. They attribute this behavior to the specificity of the triples. A compact triple, which does not contain complete clauses as part of a constituent or contain additional information, is easier to align to DBpedia. Our goal is to leverage neural-based methods to extract compact triples.

2.2 System Architecture

We focus on extraction from individual clauses within a sentence, where each clause includes a subject, a verb, optionally a direct object, and a compliment. Since extractions from different clauses share information, we split the OpenIE task into two sub-tasks: *constituent extraction* and *constituent linking*.

The task of constituent extraction is to find a set of constituents such that each constituent c is a contiguous span of words $c.span = \{(w_i, w_j)\}$ and has a pre-defined type $c.type \in Y_c$ where $Y_c = \{Argument, Predicate\}$. The constituent that takes the *relation* role in a triple has $c.type = Predicate$, and *subject* and *object* constituents have $c.type = Argument$. This schema simplifies the task and provides more information to the constituent linking model.

The task of constituent linking is to connect a given set of $Predicate$ constituents $\{p_1, \ldots p_m\}$ and $Argument$ constituents $\{a_1, \ldots a_n\}$ to obtain triples. We formulate this as a relation classification task where the set of relations is $Y_r = \{Subject, Object\}$. The model predicts relations r between each p_x and $\{a_1, \ldots a_n\}$ such that:

[1] Source code, benchmark dataset, and related resources are available at https://github.com/FarimaFatahi/CompactIE

$\exists(i,j) : r(a_i, p_x) = Subject\,,\ r(p_x, a_j) = Object$

to construct triple $(a_i; p_x; a_j)$.

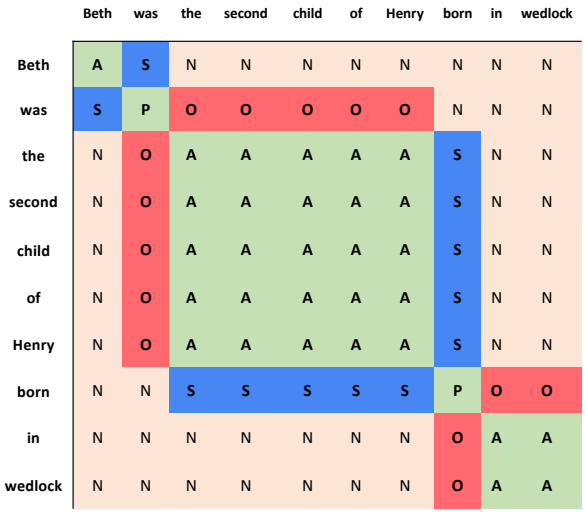

Figure 2: Table filling based on the relation between each pair of words in the sentence. Argument (**A**) and Predicate (**P**) are constituent types. Subject (**S**) and Object (**O**) declare the relation between two constituents (**N** stands for no relation).

3 Approach

In this section we describe our pipeline system, CompactIE. We first detail the constituent extraction model, its training constraints, and the decoding algorithm in Section 3.1. Then, we describe the constituent linking model in Section 3.2. Figure 3 shows an overview of CompactIE architecture.

3.1 Constituent Extraction Model

The constituent extraction model aims to find constituent spans and their types in a sentence. Following recent progress in entity-relation extraction (Wang et al., 2021), we model this as a table filling problem. However, we design a new table schema for the constituent extraction task. Figure 2 shows an example schema. A sentence s with $|s|$ tokens corresponds to a table $T^{|s|\times|s|}$ such that each cell is labeled based on the relation between the pair of words. For each constituent, corresponding cells are labeled with $y_c \in \{Argument, Predicate\}$. For relations between different constituents, corresponding cells are labeled with $y_r \in \{Subject, Object\}$. The cells with no relations are labeled $None$. Graphically, constituents are squares on the diagonal, and relations are rectangles off the diagonal.

3.1.1 Table Filling Model

Given the tabular formulation, the constituent extractor performs two tasks: a) fill the table by predicting labels for each word pair, b) extract the constituents given the label probabilities. Following (Wang et al., 2021), we adopt a biaffine attention mechanism, described next, to learn interactions between word pairs when filling the table.

Given the input sentence s, we first obtain contextual representation $\mathbf{h_i}$ for each word using a pre-trained language model (e.g. BERT (Devlin et al., 2018)). We then employ two MLPs to identify the head and tail role of the word given its vector representation h_i.

$$h_i^{head} = MLP_{head}(h_i), h_i^{tail} = MLP_{tail}(h_i)$$

Next, using the biaffine scoring function, we calculate the scoring vector of each pair of words (e.g. w_i, w_j) as follows:

$$t_{i,j} = (h_i^{head})^T U^{(1)} h_j^{tail} + (h_i^{head} \oplus h_j^{tail})^T U^{(2)} + b$$

where $U^{(1)}, U^{(2)}$ are weight parameters, b is the bias term and $\oplus$ denotes concatenation. Then, we feed the score vector t_i, j into a softmax function to calculate the probability distribution of the corresponding labels $l \in Y$, where $Y = Y_c \cup Y_r \cup None$.

$$P(y_{i,j}|s) = Softmax(t_{i,j})$$

Finally, we train the 2D table to minimize the following training objective:

$$L_{entry} = -\frac{1}{|s|^2} \sum_{i=1}^{|s|} \sum_{j=1}^{|s|} \log(P(y_{i,j} = Y_{i,j}|s))$$

where $Y_{i,j}$ is the gold label for cell (i, j) in the table.

3.1.2 Training Constraints

(Wang et al., 2021) shows that structural constraints imposed on the table during training can significantly enhance the model. We adopt their *symmetry* and *implication* constraints. However, we observed that these alone are not sufficient if certain labels are preferred over others. For example, all triples must have a subject, but some may not have an object. We propose a new *triple* constraint to further enhance our model. In this section, we describe the three constraints in detail. We also introduce $\mathcal{P} \in R^{|s|\times|s|\times|Y|}$ that denotes the stack of $P(y_{i,j}|s)$ for all word pairs in sentence s.

Symmetry: This constraint ensures that the table is symmetric i.e. the squares are symmetric about the diagonal. As shown in Figure 2, this ensures the label assigned to the (second, Henry) cell is the

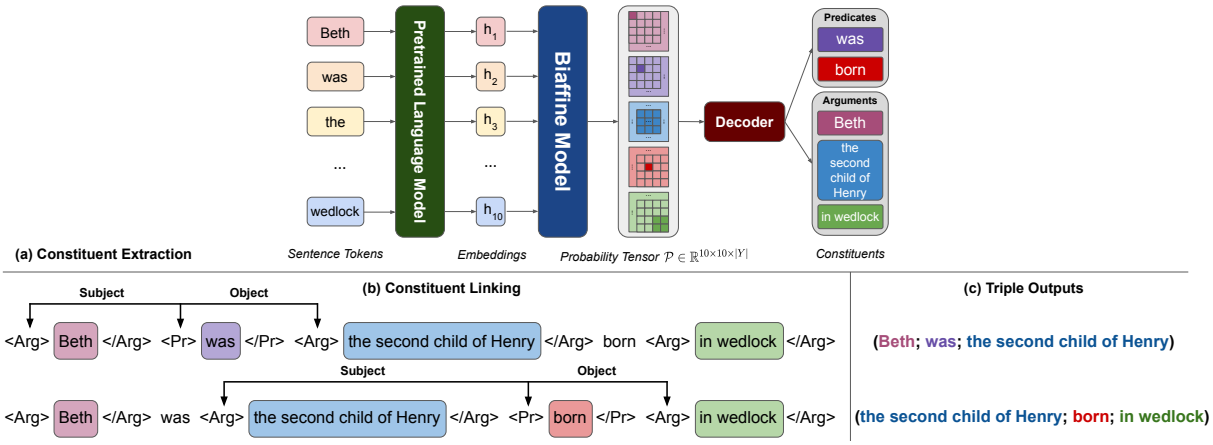

Figure 3: Overview of system architecture. Given the sentence: "Beth was the second child of Henry, born in wedlock.", the Constituent Extraction model identifies the span and type of constituents (top-right). Next, the Constituent Linking model (b) searches for $Arguments$ of each $Predicate$ constituent independently. Thus, for each of the two extracted $Predicates$, it modifies the input sentence by inserting typed constituent markers (<Arg>, </Arg> to specify the start and end of arguments and <Pr>, </Pr> for predicates). Finally, the modified sentence is fed into a classifier to find $Subject$ and $Object$ of each $Predicate$ and form triples (c).

same as the cell (Henry, second). Given matrix $\mathcal{P}$, We formulate this constraint as symmetrical loss:

$$L_{sym} = -\frac{1}{|s|^2}\sum_{i=1}^{|s|}\sum_{j=1}^{|s|}\sum_{t\in Y_r\cup Y_c}|\mathcal{P}_{i,j,t} - \mathcal{P}_{j,i,t}|$$

Implication: This constraint implies that no relation would appear unless its constituents are present in the table. This is imposed on $\mathcal{P}$: for each word in the diagonal, maximum possibility over the constituent type space $Y_c = \{Argument, Predicate\}$ is not lower than the maximum possibility for other words in the same row or column over the relation type space $Y_r = \{Subject, Object\}$.

$$L_{imp} = \frac{-1}{|s|}\sum_{i=1}^{|s|}\left[\max_{t\in Y_r}(\mathcal{P}_{i,:,t}, \mathcal{P}_{:,i,t}) - \max_{t\in Y_c}(\mathcal{P}_{i,i,t})\right]_*$$ [2]

Triple Constraint: This constraint enables the model to increase the likelihood of certain roles (e.g. ***Subject***) over the others (e.g. ***Object***) to ensure the triples are valid. We enforce this constraint on $\mathcal{P}$: For each column or row corresponding to a *Predicate* constituent, the maximum possibility of off-diagonal words over $Subject$ type is not lower than the maximum possibility of off-diagonal words over $Object$ type. We formulate this constraint as triple loss.

$$L_{triple} = \frac{-1}{2|ps|}\sum_{i\in ps}\Big[\{max(\mathcal{P}_{i,:,\boldsymbol{O}}) - max(\mathcal{P}_{i,:,\boldsymbol{S}}\} + \{max(\mathcal{P}_{:,i,\boldsymbol{O}}) - max(\mathcal{P}_{:,i,\boldsymbol{S}})\}\Big]$$

where ps is union of $Predicate$ spans in sentence.

Finally, we jointly optimize four objectives in training: $L_{entry} + L_{sym} + L_{imp} + L_{triple}$

3.1.3 Decoding

Given the label probability tensor $\mathcal{P}$, we need to decode the constituents in the testing phase. We follow a 2-step decoding procedure that finds spans of constituents first and then assigns a label to each span. The decoder first calculates the distance between adjacent rows and columns of the table to find constituents' boundaries. Next, it assigns a type to each span and filters out any $None$ constituents before passing the output to the linking model. The upper part of Figure 3 shows the output of the decoder, which extracts two constituents ("was", "born") of type $Predicate$ and three constituents ("Beth", "the second child of Henry", "in wedlock") of type $Argument$. We provide a detailed description of the decoding algorithm in Appendix A.2.

3.2 Constituent Linking Model

The constituent linking model aims to take a $Predicate$ constituent and a set of $Argument$ constituents as input and predict a relation label

[2] $[u]_* = max(u, 0)$ is the hinge loss.

$Y_r = \{Subject, Object, None\}$. This procedure is repeated for each predicate constituent in the sentence. We formulate this as a relation classification task where the model classifies relation labels of given constituent pairs based on context.

Following prior work (Zhang et al., 2019; Zhong and Chen, 2020), we modify the token sequence of input sentence by adding marker tokens <Pr>, </Pr>, <Arg>, </Arg> to highlight the constituent spans and their types. The markers help the linking model combine context information and constituent information for relation classification. As shown in Figure 3.a, two types of constituents are extracted from the input sentence. For each constituent of type $Predicate$, we modify the input sentence by highlighting the location of the $Predicate$ and all $Argument$ constituents. Then, we feed this processed sentence to a pre-trained encoder (BERT).

Next, we concatenate the output representation of the start position of predicate p with the output representation of the start position of argument a_i:

$$X_r(p, a_i) = h_{start(p)} \oplus h_{start(a_i)}$$

Finally, we feed the concatenated representation into a multi-layer perceptron (MLP) to predict the probability distribution of the relation type $r \in Y_r \cup None$:

$$P(r|p, a_i) = MLP(X_r)$$

4 Benchmark Creation

To train the constituent extraction and constituent linking models for extracting compact triples, we need a benchmark of compact triples. Existing OpenIE benchmark [3] is created by combining extractions from multiple existing OpenIE systems. Although widely adopted, we observed that it includes over-specific and sometimes incorrect extractions from previous systems. This encouraged us to design a data processing algorithm that can extract compact triples from scratch. Inspired by rule-based OpenIE system (Corro and Gemulla, 2013), we find compact triples by extracting the following clauses within a sentence:

Main Clauses are independent clauses that express a complete concept.

Complement Clauses are subordinate clauses that serve to complete the meaning of a verb or noun in the sentence.

[3] https://github.com/dair-iitd/imojie/tree/master/benchmark

	Our Benchmark			OIE2016
	Total	Train	Valid	
# Sentence	54.9k	54.5k	500	92.7k
# Triples	121.8k	120.6k	1155	190.6k
Avg. # triples per sent.	**3.165**	-	-	2.542
Avg. constituent length	**4.587**	-	-	7.893

Table 1: Statistics of our benchmark and OpenIE2016 benchmark.

Coordinate Clauses are independent clauses joined to the main clause using coordinating conjunctions such as *and*, *or*, *but*, etc.

We identify clauses within a sentence using its dependency graph. We first build a sentence tree such that the root is the head of the main clauses and the first-level children are clauses modifying the root word. We then perform a postfix traversal of the tree until we find a sub-tree with no clausal children. At this point, we run a standard OpenIE system, IMoJIE (Kolluru et al., 2020b), over the clause corresponding to the sub-tree to obtain triples. We then backtrack and extract triples for other clausal children and lastly the parent. We provide pseudo-code of algorithm in Appendix A.

We run our algorithm on each multi-clause sentence in the OpenIE2016 benchmark and obtain a new benchmark tailored for extracting compact triples. Figure 1 shows example sentences and compact triples from this benchmark.

5 Experimental Setup

Training Dataset: We train CompactIE using the benchmark described in Section 4. Table 1 compares the statistics of our new benchmark and bootstrapped OpenIE2016 benchmark. As shown, our benchmark has 1.25 times more extractions per sentence than OpenIE2016 and its constituents are more compact. We use about 1% of sentences for validation and the remaining for training.

Comparison Systems: We compare CompactIE against state-of-the-art sequence-labeling systems, OpenIE6 (Kolluru et al., 2020a) and Multi2OIE (Ro et al., 2020), and sequence-generation system, IMoJIE (Kolluru et al., 2020b)). We also compare it against traditional non-neural systems designed for extracting compact facts: NestIE (Bhutani et al., 2016) and MinIE (Gashteovski et al., 2017).

Evaluation Datasets and Metrics: We evaluate the OpenIE systems both automatically and manually on standardized benchmarks. For automatic

Dataset	Wire57		CaRB	
	Proc	Orig	Proc	Orig
# Sentences	56	57	577	641
# Triples	309	325	2101	2715

Table 2: Statistics of evaluation datasets, Wire57 and CaRB, before (Orig) and after processing (Proc).

evaluation, we first assess all systems with CaRB[4] test and Wire57 [5] datasets. Since these datasets are not targeted for compact triples, for a fair comparison we exclude triples that have at least one clause within a constituent. Table 2 shows the statistics of the original and processed datasets. Each dataset also provides its own scoring function. We report precision (P), recall (R), and F1 computed by these scoring functions. Wire57 contains more fine-grained extractions than the CaRB dataset and its scoring function is more rigorous for compact facts since it penalizes over-specific extractions. However, both CaRB and Wire57 scoring functions are based on token-level matching of system extractions against ground truth facts. Moreover, these benchmarks are incomplete, meaning that the gold extractions do not include all acceptable surface realizations of the same fact. These drawbacks encouraged us to additionally perform a fact-centered evaluation using the BenchIE (Gashteovski et al., 2021) benchmark and scoring paradigm. Finally, we carry out a manual evaluation on 100 sentences to avoid bias towards different scorers.

Implementation Details: Since the schema design of the table filling model does not support conjunctions inside constituents, we follow previous work (Kolluru et al., 2020a) and pre-process the sentences into smaller conjunction-free sentences before passing them to the system.

For a fair comparison to previous work, we use *bert-based-uncased* (Devlin et al., 2018) as the text encoder for both the constituent extraction model and constituent linking model. Each model contains nearly 110M parameters. For both models, we set the max sequence length to 512, initial learning rate to 5e-5, weight decay to 1e-5, and the batch size to 32. We use AdamW optimizer to fine-tune each model. The batch size is 300 for constituent extraction model and 20 for the constituent linking model, both equipped with early stopping. We use NVIDIA GeForce RTX 2080 Ti GPU to train both models for a cumulative time of 8 hours.

[4] https://github.com/dair-iitd/CaRB
[5] https://github.com/rali-udem/WiRe57

6 Experimental Results

6.1 Automatic Token-level Evaluation

Table 3 summarizes the performance of OpenIE systems across the CaRB and Wire57 datasets and scoring functions. On the fine-grained Wire57 dataset with a strict Wire57 scorer, COMPACTIE outperforms neural OpenIE systems (by 7.2 - 9 F1 pts) and non-neural systems (by 5.8 - 10.8 F1 pts).

On the more coarse-grained CaRB dataset, almost all OpenIE systems achieve comparable performance in terms of overall F1 using the CaRB scoring function. The neural systems still outperform non-neural systems in terms of F1, which is in line with previous studies. However, neural OpenIE systems are tuned based on the CaRB scoring function and thus tend to produce extractions that are biased towards this scoring method. Previous works (Kolluru et al., 2020a) also report issues with the scoring function not being able to handle conjunctions properly. Table 7 shows the limitations of the CaRB benchmark and scoring function through an example. As illustrated, the set of extractions produced by COMPACTIE is more exhaustive than IMoJIE and ground truth extractions. However, the CaRB scoring function assigns an F1 score of 62.0 to IMoJIE extractions, and 39.7 to COMPACTIE extractions. To resolve incompleteness of the CaRB benchmark and potential bias towards its scoring function, we undertake a fact-centered evaluation, detailed in Section 6.2, and a manual evaluation, described in Section 6.3.

6.2 Fact-centric Evaluation

(Gashteovski et al., 2021) claims that CaRB and Wire57 benchmarks and scoring functions overestimate a system's ability to extract correct facts. They propose an alternative benchmark and evaluation framework, BenchIE, that exhaustively lists all fact-equivalent extractions and clusters them into fact synsets. The scoring function considers an extraction as correct, if and only if it exactly matches any of the gold extractions from any of the fact synsets. They report Precision, Recall, and F1 based on exact triple matching.

Table 5 shows the performance of different OpenIE systems on BenchIE. As shown, COMPACTIE outperforms all other systems except MinIE. We found that MinIE aims to exhaustively produce different representations of the same fact. In contrast, COMPACTIE follows the setup of neural OpenIE systems and encourages at most one repre-

System	Wire57						CaRB					
	P	R	F1	ACL	NCC	RPA	P	R	F1	ACL	NCC	RPA
NestIE	35.0	15.0	21.0	**4.65**	0.07	1.16	**53.4**	32.8	40.6	**4.29**	0.08	1.21
MinIE	31.3	30.7	31.0	4.93	0.2	**1.6**	35.3	**50.5**	41.6	4.97	0.4	**1.57**
IMoJIE	41.2	20.1	27.0	6.23	0.26	1.07	48.5	44.6	**46.5**	6.43	0.39	1.08
OpenIE6	27.7	19.4	22.8	5.98	0.66	1.14	44.3	44.5	44.4	6.26	0.56	1.29
Multi2OIE	33.4	18.9	24.1	5.54	0.42	1.05	48.2	44.5	46.3	6.06	0.42	1.08
CompactIE	**41.4**	**25.8**	**31.8**	5.23	**0.05**	**1.37**	51.3	39.9	45.0	5.08	**0.07**	**1.32**

Table 3: Performance of OpenIE systems on Wire57 and CaRB datasets. The three analytic metrics (ACL, NCC, RPA) are discussed in Section 7.

System	Precision	Compactness
NestIE	49.1 (84/171)	98.8 (83/84)
MinIE	58.0 (217/374)	78.8 (171/217)
IMoJIE	90.0 (156/173)	53.2 (83/156)
OpenIE6	78.0 (210/269)	65.2 (137/210)
Multi2OIE	78.6 (151/192)	59.6 (90/151)
CompactIE	75.8 (175/231)	94.9 (166/175)

Table 4: Manual evaluation of OpenIE systems on CaRB validation set. Precision indicates the percentage of correct extractions. Compactness indicates the percentage of compact extractions amongst the correct ones.

System	BenchIE		
	P	R	F1
NestIE	37.1	10.2	16.0
MinIE	**42.9**	**27.8**	**33.7**
IMoJIE	34.3	12.8	18.6
OpenIE6	31.1	**21.4**	25.3
Multi2OIE	39.2	16.1	22.8
CompactIE	**40.3**	19.0	**26.2**

Table 5: Performances of OpenIE systems on the BenchIE dataset.

sentation per fact. As a result, MinIE produces 1.36x more extractions than CompactIE, achieving much higher recall than its neural counterparts.

6.3 Manual Evaluation

Limitations in the aforementioned benchmarks and evaluation frameworks encouraged us to perform human evaluation on triples generated by various systems. To this end, we randomly select 100 sentences from the CaRB validation set and feed them to all systems to investigate the generated triples. Next, we ask two graduate CS students, blind to the OpenIE systems, to mark each triple for correctness (0 or 1) based on whether it is asserted in the text and correctly captures the semantic information. They also label extractions for compactness (0 or 1). We consider an extraction compact if none of its constituents is longer than 10 words, includes conjunction or can be an independent extraction. We found an inter-annotator agreement of 0.68 on correctness and 0.83 on compactness using the Cohens Kappa metric. We report the results of the manual evaluation in Table 4. Neural systems target informativeness, which results in high precision at the cost of compactness. On the other hand, non-neural systems that aim for compact triples suffer from low precision. CompactIE offers a better trade-off between precision and compactness. It achieves comparable precision to neural models (-6 %) while providing substantially more compact extractions (+36 %). Compared to the MinIE, CompactIE produces triples with significantly higher precision (+22 %) while producing a comparable number of compact triples. NestIE achieves comparable compactness rate to CompactIE but suffers from low precision and total number of extractions.

7 Analysis

7.1 Compact and Overlapping Constituents

To understand the performance of CompactIE in generating compact triples that share constituents, we introduce the following metrics:

- Average Constituent Length (ACL): average length of constituents across all system-generated triples. This is a "syntactic" measure of compactness. The lower the ACL score, the higher the compactness of triples.
- Number of Constituent Clauses (NCC): average number of clauses per constituent that could be extracted as independent triples. The lower the NCC score, the better the compactness of triples.
- Repetitions Per Argument (RPA): number of total arguments divided by the number of unique arguments. The higher the RPA score, the higher fraction of total constituents produced per sen-

tence are shared.

Table 3 summarizes the performance on these metrics over CaRB and Wire57 benchmarks. We do not conduct a separate analysis over BenchIE since it uses a subset of CaRB sentences. As shown, the ACL scores of CompactIE are significantly lower than its neural counterparts and closely follows MinIE. The average constituent length (ACL) of NestIE triples is the lowest since it breaks sentences into small triples with verb, noun, preposition, and adjective mediated relations. For instance, the sentence: "2 million people died of AIDS." is broken down into T1: (2 million people; died), and T2: (T1; of; AIDS). However, its fine-grained strategy greatly sacrifices F1 for compactness. CompactIE achieves the lowest NCC score which indicates that the constituents in triples contain the fewest verbal clauses. As a result, these triples are more suitable for downstream applications such as text summarization and knowledge-base construction than other counterparts.

Finally, high RPA scores of CompactIE demonstrate the effectiveness of our approach as it enables the system to reuse the same constituent to generate multiple triples. MinIE achieves a slightly higher RPA score than CompactIE since it extracts multiple triples to represent the same fact leading to a higher repetition of unique constituents.

7.2 Effectiveness of Design Choices

Pipelined Approach vs. Unified Table Filling. To compare our pipelined approach with a unified extraction model, we follow UniRE (Wang et al., 2021), which decodes a single table to identify entities and relations jointly. We follow their 3-step decoding algorithm to obtain the constituents and links between them from the same table (with the schema shown in Figure 2). We refer to this model as CompactIE$_{\text{uni}}$. We report the performances in Table 6 and show that performance drops by jointly training the constituent and linking model. This aligns with the observations in recent entity-relation extraction work that pipelined approaches are more effective than joint models.

Effectiveness of Schema Design. Our table schema for constituent extraction includes both labels for constituents as well as labels to link them. We argued that this design captures the contextual dependency information between the constituents that boosts extraction performance. We compare the effectiveness of this schema design to another schema that uses only constituent labels $Y_c : \{Argument, Predicate\} \cup None$. Note that we use the same constituent linking model to obtain triples from the extracted constituents. We refer to this setting as CompactIE$_{\text{const table}}$. Table 6 illustrates the performance of this system on both CaRB and Wire57 datasets. We find that CompactIE achieves significantly higher F1 compared to CompactIE$_{\text{const table}}$ and conclude that incorporating additional context in the table schema improves the performance of the constituent extraction model.

Method	Wire57	CaRB
CompactIE	31.8	45.0
CompactIE$_{\text{uni}}$	17.6	35.8
CompactIE$_{\text{const table}}$	26.0	40.1

Table 6: Comparing F1 scores of CompactIE against joint extraction systems.

7.3 Error Analysis

We examine CompactIE triples produced for 50 randomly selected sentences of the CaRB validation dataset and 20 randomly selected sentences of the Wire57 dataset. Upon close analysis, we identify five major sources of error:

Constituent Not Found: (49.29%) We find that the constituent extraction model can fail to correctly label the constituents in the table. We found that the model gets biased towards producing $None$ labels due to the imbalanced distribution of labels.

Wrong Relation Type: (28.17%) These involve errors where the constituent linking model fails to correctly predict the link between the constituents. The current model encodes one sentence per predicate to find its arguments. Alternatively, we can encode one sentence per predicate-argument pair to focus more on each relation. Relation labels in the constituent extraction model can also assist the linking model in predicting the correct relations. We reserve this issue for future work.

Boundary Detection Error: (11.26%) These include errors where the decoder in constituent extraction fails to correctly identify the boundaries of the constituents. Boundary detection in constituent extraction model is highly dependant on the choice of distance threshold (α), as explained in A.2, which limits its robustness.

Inexpressive Table Error: (7.04%) These include errors where constituents have overlapping spans that participate in two roles within the same extrac-

System	Subject	Predicate	Object	F1
Gold	Applications	use this service to record	activity for a system	-
	other OSIDs	use the service to record	data	
	other OSIDs	use the service to record data	during analysis	
IMoJIE	Applications	use this service to record	activity for a system	62.0
	other OSIDs	use	the service to record data during ... analysis	
CompactIE	Applications	use	this service to record activity for a system	39.7
	other OSIDs	use	service to record data during development	
	other OSIDs	use	the service	
	the service	record	data during debugging	
	the service	record	data during analysis	

Table 7: Gold, IMoJIE and CompactIE extractions for the sentence: "*Applications use this service to record activity for a system while other OSIDs use the service to record data during development, debugging, or analysis.*" and their CaRB F1 score that evaluates extractors triples against gold triples.

tion or two different extractions.

Less than 4.22% of the errors were because of incorrect constituent type predictions. This indicates the effectiveness of our table filling method on constituent type detection.

8 Related Work

OpenIE has been studied extensively for over a decade with a history of statistical and rule-based systems (Banko et al., 2007; Fader et al., 2011; Corro and Gemulla, 2013; Mausam et al., 2012; Angeli et al., 2015) that extract triples from sentences without using any training data. Recently, neural models have been developed that are trained end-to-end on extractions bootstrapped from previous OpenIE systems. These can broadly be classified into *labeling-based* and *generation-based* systems.

Labeling-based systems (Stanovsky et al., 2018; Kolluru et al., 2020a; Ro et al., 2020) tag each word in the sentence and construct triples in an auto-regressive manner or by using a unique predicate for each triple. Generation-based systems (Kolluru et al., 2020b; Bhutani et al., 2019) use a sequence-to-sequence model to generate triples one word at a time. Labeling-based systems can handle redundancy in extracted triples and are faster than generation-based systems (Kolluru et al., 2020a).

Compactness in OpenIE: There has been prior work (Bhutani et al., 2016; Gashteovski et al., 2017; Stanovsky and Dagan, 2016; Angeli et al., 2015) that focuses on finding compact triples and shows that concise triples are useful in several semantic tasks. However, recent studies (Léchelle et al., 2018; Gashteovski et al., 2020) indicate that neural OpenIE systems produce more specific triples with additional information than conventional OpenIE systems and are harder to align with existing knowledge bases. Therefore, we focus on designing a neural OpenIE system that extracts compact triples.

Grid Labeling: Also known as table filling, grid labeling has been recently applied to entity relation extraction (Gupta et al., 2016; Wang et al., 2021) and open information extraction tasks (Kolluru et al., 2020b). However, these models map entities (constituents) and relations (subject, object) in a unified label space to capture the inter-dependency between them. (Zhong and Chen, 2020) shows that a pipelined approach for entity and relation extraction outperforms prior joint models that use the same encoder for the two sub-tasks. In this work, we validate this claim for the OpenIE task. Furthermore, we design a grid labeling schema that identifies constituents and their types, akin to entities in the entity relation extraction task.

9 Conclusion

In this work we extract compact triples from single sentences using an end-to-end pipelined approach, first extracting triple constituents using a novel table filling model and then determining relations between them with a classifier. Our method achieves excellent performance in producing exhaustive compact triples with high precision. We hope that CompactIE serves as a strong baseline and makes us re-think the value of all-at-once information extraction systems.

10 Acknowledgments

The research described herein was sponsored by the U.S. Army Research Institute for the Behavioral and Social Sciences, Department of the Army (Contract No. W911NF-20-C-0028). The views expressed in this presentation are those of the author and do not reflect the official policy or position of the Department of the Army, DOD, or the U.S. Government.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proc. ACL-IJCNLP '15*, pages 344–354.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proc. IJCAI '07*, page 2670–2676, San Francisco, CA, USA.

Nikita Bhutani, H. V. Jagadish, and Dragomir Radev. 2016. Nested propositions in open information extraction. In *Proc. EMNLP '16*, pages 55–64, Austin, Texas. Association for Computational Linguistics.

Nikita Bhutani, Yoshihiko Suhara, Wang-Chiew Tan, Alon Halevy, and HV Jagadish. 2019. Open information extraction from question-answer pairs. In *Proc. NAACL-HLT '19*, pages 2294–2305.

Luciano Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *Proc. WWW '13*, 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.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proc. EMNLP '11*, pages 1535–1545.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale seq2seq models to multi-document inputs. *CoRR*, abs/1910.08435.

Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. 2017. MinIE: Minimizing facts in open information extraction. In *Proc. EMNLP '17*, pages 2630–2640, Copenhagen, Denmark.

Kiril Gashteovski, Rainer Gemulla, Bhushan Kotnis, Sven Hertling, and Christian Meilicke. 2020. On aligning openie extractions with knowledge bases: A case study. In *Proc. EMNLP '20 Workshop on Evaluation and Comparison of NLP Systems*, pages 143–154.

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

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proc. COLING '16*, pages 2537–2547.

Zengguang Hao, Binxia Xu, Shiyuan Zheng, and Yang Gao. 2018. Structured text summarization via open domain information extraction. In *Proc. CSCWD '18*, pages 701–706. IEEE.

Heng Ji, Benoit Favre, Wen-Pin Lin, Dan Gillick, Dilek Hakkani-Tur, and Ralph Grishman. 2013. Open-domain multi-document summarization via information extraction: Challenges and prospects. In *Multi-source, multilingual information extraction and summarization*, pages 177–201. Springer.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. *CoRR*, abs/1704.05572.

Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, Mausam, and Soumen Chakrabarti. 2020a. Openie6: Iterative grid labeling and coordination analysis for open information extraction.

Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Mausam, and Soumen Chakrabarti. 2020b. Imojie: Iterative memory-based joint open information extraction.

William Léchelle, Fabrizio Gotti, and Philippe Langlais. 2018. Wire57: A fine-grained benchmark for open information extraction. *arXiv preprint arXiv:1809.08962*.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proc. EMNLP '12*, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.

Mausam Mausam. 2016. Open information extraction systems and downstream applications. In *Proc. IJCAI '16*, page 4074–4077.

Youngbin Ro, Yukyung Lee, and Pilsung Kang. 2020. Multi2oie: Multilingual open information extraction based on multi-head attention with bert. *arXiv preprint arXiv:2009.08128*.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proc. EMNLP '16*, pages 2300–2305, Austin, Texas.

Gabriel Stanovsky, Ido Dagan, et al. 2015. Open ie as an intermediate structure for semantic tasks. In *Proc. ACL-IJCNLP '15*, pages 303–308.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proc. NAACL '18*, pages 885–895.

Yijun Wang, Changzhi Sun, Yuanbin Wu, Hao Zhou, Lei Li, and Junchi Yan. 2021. Unire: A unified label space for entity relation extraction.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.

Zexuan Zhong and Danqi Chen. 2020. A frustratingly easy approach for joint entity and relation extraction. *CoRR*, abs/2010.12812.

A Appendix

A.1 Benchmark Creation

The Algorithm 2 gives a high-level overview of our benchmark creation mechanism while a lot of details and difficulties have been omitted. The Benchmark Creation Algorithm extracts triples for each sentence using the Algorithm 1. The OpenIE system used to produce triples out of simple clauses is IMoJIE (Kolluru et al., 2020b).

The following example illustrates the benchmark creation algorithm. Given the sentence: *"The group reach a small shop, where the crocodile breaks through a wall"*, the algorithm first builds the sentence tree as shown in Figure 4. Then, starting from the root, *ExtractTriple* function traverses the tree until it reaches a child ("breaks") with no further clausal children. At this point, a clause for the subtree rooted at "breaks" is generated and fed into the IMoJIE system. IMoJIE extracts triple: (the crocodile; breaks; through a wall) out of this clause. Then, since both children of the root ("reach") are processed, the IMoJIE triple of the root's corresponding clause is extracted as (The rest of the group; reach; a small shop).

Algorithm 1: ExtractTriples

```
Data: Tree Node R
Result: Set of compact triples T
T = set() ;
for child in R.children do
    if child has no clausal child then
        T += IMoJIE(child.clause) ;
    end
    else
        T+= ExtractTriples(child) ;
    end
end
T += IMoJIE(R.clause) ;
return T
```

Algorithm 2: Benchmark Creation

```
Data: Sentence List S = [s_1, s_2, .., s_n]
Result: B benchmark of compact triples for
        sentences in S
B = set() ;
for sentence in S do
    root = build_sentence_tree(sentence) ;
    B += ExtractTriples(root) ;
end
return B
```

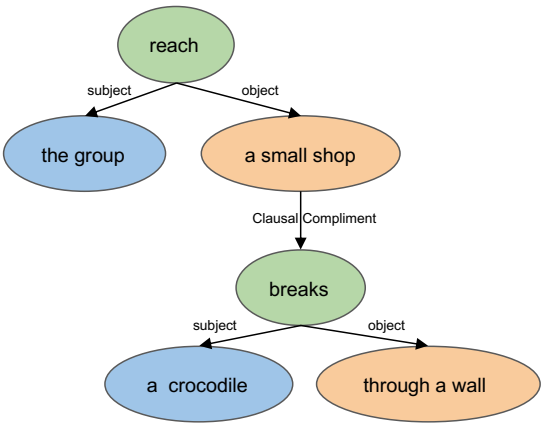

Figure 4: Sentence Tree for input sentence: *"The group reach a small shop, where the crocodile breaks through a wall"*.

A.2 Table Decoding

Following the (Wang et al., 2021) work, in the testing phase, we rely on the label probability tensor $\mathcal{P} \in \mathbb{R}^{|s|\times|s|\times|Y|}$ of the sentence s, to first extract constituent spans, and then predict the constituent type. Next, we describe the decoding procedure.

A.2.1 Constituent Span Detection

One important observation of the ground truth table is that a constituent's corresponding rows and columns are identical (e.g., row 2 and row 3 of Figure 2 are identical). Therefore, given the tensor $\mathcal{P}$, we compute the distance of adjacent rows (and columns). If the distance is larger than a predefined threshold α (which is set to 1.2), a split position is detected. This means that the two adjacent rows (columns) belong to different constituents or one belongs to a constituent while the other is not. Following the (Wang et al., 2021) work, we flatten the $\mathcal{P}$ tensor from both row and column perspectives and calculate the euclidean distance of adjacent rows and adjacent columns. Finally, we average these two distances as the final distance and compare the final distance to α to find the span of different constituents.

A.2.2 Constituent Type Detection

Given a constituent's span (i, j), we decode the constituent type $t^* \in Y$, where $Y = Y_c \cup Y_r \cup None$, according to its corresponding square symmetric about the diagonal:

$$t^* = argmax_{t\in\{Y_c\cup None\}} Avg(P_{i:j,i:j,t})$$

Spans with predicted type $t^* \in Y_c$ are regarded as constituents and passed to the constituent linking model.