MILIE: Modular & Iterative Multilingual Open Information Extraction

Anonymous ACL submission

Abstract

Open Information Extraction (OpenIE) is the task of extracting *(subject, predicate, object)* triples from natural language sentences. Current OpenIE systems extract all triple slots independently. In contrast, we investigate the hypothesis that it may be beneficial to extract triple slots iteratively: first extract easy slots, followed by the difficult ones by conditioning on the easy slots, and therefore achieve a better overall extraction.

Based on this hypothesis, we propose a neural OpenIE system, MILIE, that operates in an iterative fashion. Due to the iterative nature, the system is also modular—it is possible to seamlessly integrate rule based extraction systems with a neural end-to-end system, thereby allowing rule based systems to supply extraction slots which MILIE can leverage for extracting the remaining slots. We confirm our hypothesis empirically: MILIE outperforms SOTA systems on multiple languages ranging from Chinese to Arabic. Additionally, we are the first to provide an OpenIE test dataset for Arabic.

1 Introduction

Open Information Extraction (OpenIE) aims to extract structured facts in the form of *(subject, relation, object)*-triples from natural language sentences (Etzioni et al., 2008). For example, given a sentence, *"Barrack Obama became the US President in the year 2008"*, an OpenIE system is expected to extract the following triples: (*Barrack Obama*; *became*; *US President*) and (*Barrack Obama*; *became US President in*; *2008*). We refer to subject, predicate and the object of the triple as slots of a triple. OpenIE extractions are human understandable intermediate representations of facts in source texts (Mausam, 2016), which are useful in a variety of information extraction end tasks such as summarization (Xu and Lapata, 2021), question answering (Khot et al., 2017; Yan et al., 2018) and automated schema extraction (Nimishakavi et al., 2016).

The various slots of a triple are dependent on each other and hence an error in one slot renders the entire extraction unusable. We hypothesize that triple extraction errors largely stem from the difficulty of extracting certain slots of triple and these difficult to extract slots may depend on the sentence construction and also the language. For example, *"Barrack Obama became the US President in the year 2008"* contains two triples *(Barrack Obama; became; US President)* and *(Barrack Obama; became US President in; 2008)*. Extracting the predicate, *"became US President in"*, for the second triple is tricky, because the object of the first triple (US President) overlaps with the predicate of the second triple. But if the extraction system was provided with the object, *(2008)*, and then asked to extract a triple conditioned on this object, then the predicate extraction would be easier.

This is precisely the hypothesis we wish to investigate —is it easier to extract certain slots of a triple, say subjects, compared to other slots, such as objects, and is it possible to improve performance by leveraging specific slot extraction orders?

Given the hypothesis, we propose MILIE, a Modular & Iterative multiLingual open Information Extraction system, which iteratively extracts the different slots of a triple. The iterative nature allows for (1) studying the effects of a slot extractions on rest of the triple extraction, (2) extracting easier triple slots followed by harder ones, (3) aggregating different slot extraction orders as a mixture of experts, and (4) integrating slots supplied by an external rule-based system, resulting in a hybrid system. The latter offers a system that combines the best of neural and rule based systems, e.g. by using a rule-based system to extract high precision slots on which the neural system is conditioned.

We empirically confirm our hypothesis: the iterative nature of MILIE outperforms several SOTA

systems. It proves especially useful for zero-shot multilingual extraction, which we evaluated on five different low resource languages. Additionally we show how MILIE can leverage rule-based slot extraction by conditioning on them to predict the remaining parts of the triple. Therefore MILIE is a boon for existing applications wishing to transition from a rule based information extraction system to a neural one, because MILIE would allow using the rule-based system to compensate for the lack of exhaustive training data. Finally, we perform linguistic analyses that uncovers useful insights on how different languages either make it easy or difficult for OpenIE systems to extract individual elements of the triple.

Our contributions are summarized as follows:

1. We propose MILIE, a multilingual OpenIE system that iteratively extracts the different slots of a triple.

2. We carry out extensive experiments on a variety of languages including English and demonstrate that MILIE outperforms recent SOTA systems by a wide margin, especially on languages other than English.

3. We perform an extensive analysis based on ablation studies and uncover interesting insights about the nature of OpenIE task in different languages.

2 MILIE

The backbone of our system is the iterative procedure (Section 2.1), which allows us to investigate our hypothesis. The iterative procedure allows us to extract triple slots in various pathway orders, which results in a series of possible aggregation schemes (Section 2.2). To create a strong iterative system, the training paradigm (Section 2.3 needs to consider two aspects: (1) it needs to prepare incomplete triple extractions which represent incomplete triple extractions the system is expected to predict; (2) it creates negative samples that allow for teaching the system when to not continue with an extraction due to a prior error. With the iterative nature we also integrate rule-based systems (Section 2.4) as well as elegantly handle the specific case of n-ary extractions, where more than 3 slots need to be extracted (Section 2.5).

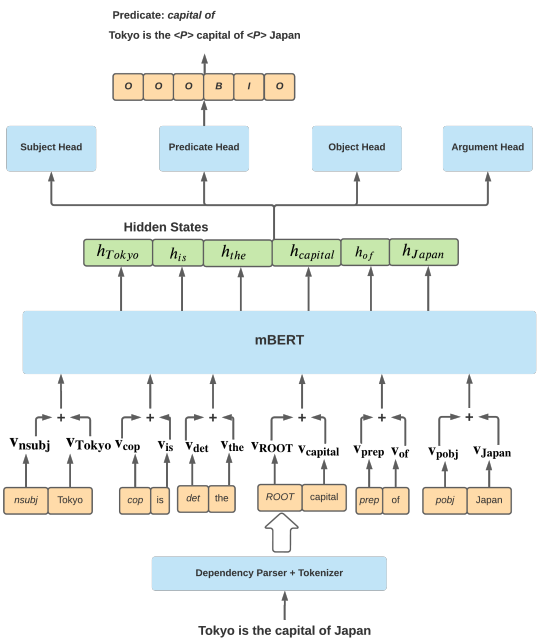

Figure 1: MILIE system architecture. An input sequence is is tokenized and, optionally, dependency parsed. This is given to a BERT-based transformer, which outputs a hidden state for each token. The hidden states are given to each of the extraction heads, here to the predicate head. This head marks the location of the predicate in the sequence. The system then proceeds to extract the other slots, see Figure 2.

2.1 Iterative Prediction

To implement the iterative nature of our system, we use a BERT-based transformer (Devlin et al., 2019) as the base building block. On top of this block, we add a total of four neural networks blocks in parallel, which we refer to as heads and which are each in charge of extracting a particular triple slot. Concretely, we have the heads f_s, f_o, f_p, f_a, which are in charge of predicting *s*ubject, *o*bject, *p*redicate and *a*rgument, respectively. The argument head is an extra feature, which is needed for n-ary extractions that occur in some datasets, where in addition to the triple there might be an argument that modifies the triple, e.g., a temporal phrase.

Given an input sequence of words of length N, $S = w_i, \ \ i = 1 \cdots N$, the task for each extraction head is framed as a BIO tagging problem. For this, each output head outputs a label l_i for token w_i, where $l_i \ \in \{B, I, O\}, \ i = 1 \cdots N$ (see Fig, 1 for the architecture). The output heads use the final transformer hidden state and predict labels denoted by L_s, L_o, L_p, L_a where $L_{(.)} = l_1, l_2, \ \cdots l_N$.

By having different extraction heads, we identify extraction slots iteratively. During prediction

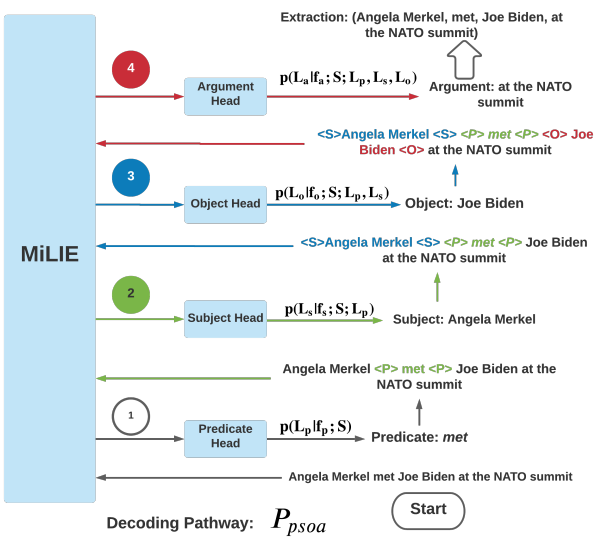

Figure 2: Iterative extraction dynamics for decoding pathway P_{psoa}. The numbers indicate the iteration number. Iterations are color coded, black is the predicate extraction, green subject extraction, blue object extraction and red argument extraction.

time, along with the input sentence, the model also expects extractions predicted by the previous iterations. To provide this information we add special symbols to the sentence that explicitly mark the previous extractions in the sentence. For example, we surround the predicate with the symbol <*P*>, subject with <*S*> and object with <*O*>. For example, for predicting the object given the predicate extracted from previous iteration, the extracted predicate is marked in the sentence using the <*P*> symbol and the sentence is consequently passed through the transformer for predicting the object using the object head. We always extract the arguments at the last iteration, therefore we do not mark the arguments in the sentence.[1]

Finally, we add the option to attach a dependency tag t_i to each word w_i in the sequence. This additional information may allow the system to more effectively learn how to extract triples. We use a language specific dependency tagger for obtaining the tags. We target languages, which are low resource for OpenIE, but could be high resource for other tasks, such as PoS tagging or dependency parsing. For a graphical overview of the MILIE architecture, see Figure 1.

[1]Preliminary experiments suggested that predicted the argument last leads to better overall results. This makes sense intuitively, as the argument can modify the entire triple.

2.2 Aggregating Decoding Pathways

The order in which the different triple parts are extracted can be varied. This allows us to investigate the challenge of extracting triple elements in specific order on different languages. Additionally different pathways aid different kinds of extractions and combining them results in a richer set of extractions. Choosing a particular order defines a decoding pathway P_{uvxy} as a sequence of output heads where $u, v, x, y \in \{s, p, o, a\}$. For example, the decoding pathway P_{spoa} denotes a sequence of output functions (f_s, f_p, f_o, f_a).

Fixing the n-ary argument extraction in the final iteration we obtain the following six decoding pathways- $P_{spoa}, P_{sopa}, P_{psoa}, P_{posa}, P_{ospa}, P_{opsa}$. Let's assume the decoding pathway P_{psoa}: predicates are extracted first, then for each predicate, subjects are extracted, then for each (predicate, subject) pair objects are extracted and finally for every extracted (predicate, subject, object) tuple all the n-ary arguments are extracted. This extraction procedure preserves the relationships between the extracted elements resulting in correctly extracting multiple triples. Fig. 2 illustrates this procedure.

We hypothesize that some triples are easier to predict if, e.g., the predicate is extracted first while for others subject first would work well. This could differ from triple to triple, but also with different languages. Consequently, some decoding pathways might be more error prone than others. This leads to two questions: (1) Which pathways are best? (2) Can we improve recall by aggregating triples using different decoding pathways?

We propose a simple algorithm we term as *Water Filling* (WF) for aggregating the extractions. This is inspired by the power allocation problem in the communication engineering literature (Kumar et al., 2008). Imagine a thirsty person with access to different pots of water with varying levels of purity and with the caveat that the amount of water is inversely proportional to the purity. The natural solution is to first drink the high purity water and move on to the pots in decreasing level of purity until the thirst is quenched. We use the same idea. Treating each decoding pathways as an expert, we assume that the triples extracted by all 6 pathways are more accurate compared to those extracted by only 5 pathways, 4 pathways and so on. This can be thought of as triples obtaining votes from experts. Starting with an empty set, for each sentence we start adding triples to the set in the

order of decreasing number of received votes. The normalized votes a triple receives is used as the confidence value of the triple. Although the procedure is explained in a sequential manner it can be parallelized by running all 6 pathways in parallel.

2.3 Training

Triple preparation. For effectively extracting different triple slots conditioned on other slots, the model needs to see such combinations during training. However, enumerating all possible combinations exhaustively is prohibitively expensive. We propose a sampling technique that ensures that the model sees varied combinations of different targets and prior extractions. This is done by creating a training set that simulates a prior extraction and forces the model to predict the next extraction. To ensure that the training dataset size does not explode, we randomly sample one pathway order for each training instance.

Based on the sampled pathway, we randomly sample at which step in the decoding process we are at and then mark the slots prior to this step in the sentence and use the remaining steps as target labels. We allow for multiple instances of the target labels, however there is only one instance of the marked element. For example, given one subject the target could be multiple predicates. This procedure trains the model to predict an appropriate label conditioned on a variety of previous predictions.

Given that triples are at different steps in their decoding process, we minimize different log-likelihood functions. We describe the log likelihood functions along with a few example of the training instances in Table 1. We list additional details in Appendix A.

Negative Sampling. Iterative prediction is prone to error amplification, i.e. if an error is made during the first iteration then the error propagates and affects subsequent extractions. Anticipating this, we train MILIE to recognize extraction errors made in the previous iteration. We purposely augment the training data with corrupted data points containing incorrectly marked extractions. For each of the incorrect extractions the model is trained to predict a blank extraction, i.e., predicting the outside label for all tokens. We use a similar sampling procedure as described previously. For every training data point from a fixed number of training data points, we create one negative sample using one of the three techniques and then choose k negative samples, where k is a hyperparameter.

We corrupt triples using three techniques: (1) corrupting the predicates by replacing them with randomly chosen tokens from the sentence, (2) corrupting the subject and object by exchanging them, and (3) by mismatching the subject object pairs from different triples. We detail the entire procedure in Appendix A.

2.4 Integrating Linguistic Rule based systems

Crucially, each output head is conditioned on the input and the output labels extracted by the previous function. This feature allows MILIE to seamlessly integrate rule based systems with neural systems since the conditioning can be also done on extractions obtained from rule based systems. This is advantageous in situations where a linguistic rule based system works well, for say, extracting objects. Then MILIE can complete the missing parts of the triple conditioned on the objects.

We treat the output of the rule based system as potential objects paired with subjects and extract the predicate connecting them. If the rule based extraction is incorrect, then MILIE can detect the error and extract nothing. This results in more accurate extractions compared to simply post-processing the extracted tokens using linguistic rules.

2.5 Binarizing n-ary Extractions

We evaluate MILIE on both n-ary as well as binary triple extraction datasets. One simple way to convert the n-ary extractions to binary extraction is to ignore the n-ary arguments. However, this will lead to a decrease in recall because the n-ary arguments may not be part of other extracted triples due to the initial n-ary extraction. Another method is to treat the extracted n-ary arguments as objects to the same subject, predicate pair. This would ensure that the extracted arguments are not dropped, however this may result in drop of precision since the n-ary argument may not attach to the same predicate. For example, consider the extraction (*Barrack Obama; became; US President; in the year 2008*). Treating n-ary arguments as objects results in (*Barrack Obama; became; US President*) and (*Barrack Obama; became; in the year 2008*) resulting in an incorrect extraction.

In contrast to the above subpar solutions, the iterative nature of MILIE allows us to elegantly address the problem of converting n-ary extractions into a binary format: we treat the extracted n-ary arguments as hypothesized objects. We then provide the

Likelihood function	Input Sentence	Head	Target
$\mathcal{L}_p = -\sum_{i=1}^{N} log\ p(l_i^p \| f_p(\theta); S)$	The Taj Mahal was built by Shah Jahan in 1643	Predicate	built by
$\mathcal{L}_s = -\sum_{i=1}^{N} log\ p(l_i^s \| f_s(\theta); S; L^p)$	The Taj Mahal was <P>built by<P> Shah Jahan in 1643	Subject	Taj Mahal
$\mathcal{L}_o = -\sum_{i=1}^{N} log\ p(l_i^o \| f_o(\theta); S; L^p; L^s)$	The <S>Taj Mahal<S> was <P>built by<P> Shah Jahan in 1643	Object	Shah Jahan
$\mathcal{L}_a = -\sum_{i=1}^{N} log\ p(l_i^a \| f_a(\theta); S; L^p; L^s; L^o)$	The <S>Taj Mahal<S> was <P>built by<P> <O>Shah Jahan<O> in 1643	Argument	in 1643
$\mathcal{L}_p = -\sum_{i=1}^{N} log\ p(l_i^p \| f_p(\theta); S; L^s; L^o)$	The <S>Taj Mahal<S> was built by <O>Shah Jahan<O> in 1643.	Predicate	built by
$\mathcal{L}_s = -\sum_{i=1}^{N} log\ p(l_i^s \| f_s(\theta); S; L^o)$	The Taj Mahal was built by <O>Shah Jahan<O> in 1643.	Subject	Taj Mahal
$\mathcal{L}_o = -\sum_{i=1}^{N} log\ p(l_i^o \| f_o(\theta); S)$	The Taj Mahal was built by Shah Jahan in 1643.	Object	Shah Jahan

Table 1: A few examples of training inputs and corresponding log likelihood functions.

English	(Ro et al., 2020) Translation	Error Explanation
The stock pot should be chilled and the solid lump of dripping which settles when chilled should be scraped clean and re-chilled for future use.	La olla de caldo debe ser enfriado y la masa sólida de goteo que se asienta cuando [se] enfriada se debe ~~raspar limpio~~ y re-enfriada para uso futuro.	*"enfriado"*: the gender of the adjective doesn't match the noun. "[se]": missing reflexive particle. *"enfriada"*: wrong use of the participle. "~~*raspar limpio*~~": syntactic error.
However, StatesWest isn't abandoning its pursuit of the much-larger Mesa.	Sin embargo, StatesWest no abandona su búsqueda de la ~~tan - Mesa grande~~.	~~<tan - Mesa grande>~~: syntactically and semantically incorrect.
The rest of the group reach a small shop, where Brady attempts to phone the Sheriff, but the crocodile breaks through a wall and devours Annabelle.	El resto del grupo llega a una pequeña tienda, donde Brady intentos ~~de teléfono del Sheriff~~, pero ~~los saltos de cocodrilo a través de una pared~~, y devora a Annabelle.	*"intentos"*: number and the gender don't match with the noun. ~~*"de teléfono del Sheriff"*~~: telefóno cannot be used as a verb. ~~*"los saltos de cocodrilo a través de una pared"*~~: semantically incorrect.

Table 2: Examples of incorrectly translated sentences. Using red we highlight mistranslated words, using blue, missing words, and with a ~~strikethrough~~ the parts that are semantically or syntactically incorrect.

extracted subject, hypothesized object pair to the model, which then extracts a new predicate conditioned on the previously extracted subject and the hypothesized object, i.e., $p(L_p \mid f_p(\theta); S; L_s =$ "Barrack Obama"; $L_o =$ "year 2008"). This creates a possibility of extracting the correct predicate, something that is not possible with existing n-ary OpenIE systems.

3 Experiments

3.1 Setup

Baselines & Training. We compare MILIE with both unsupervised and supervised baselines. Specifically we compare MILIE with ClausIE, MinIE, Stanford-OIE, RNN-OIE, OIE6 (Del Corro and Gemulla, 2013; Gashteovski et al., 2017; Stanovsky et al., 2018; Angeli et al., 2015; Kolluru et al., 2020a) and Multi2OIE (Ro et al., 2020) on English. Multi2OIE is the only neural system capable of extracting triples from multiple languages and therefore it is the only available baseline for the non-English evaluations.

We use the English RE-OIE2016 (Zhan and Zhao, 2020) training dataset used in (Ro et al., 2020). This training dataset contains n-ary extractions allowing MILIE to be evaluated on both n-ary as well as binary extraction benchmarks. Evaluation on languages other than English is always zero-shot, i.e., the model is trained using only the English Re-OIE2016 dataset and tested on test set of the other languages.

CaRB benchmark. We use the CARB benchmark introduced in (Bhardwaj et al., 2019) for evaluating English OpenIE n-ary extraction. However, the CARB benchmark also suffers from serious shortcomings due to its evaluation method based on token overlaps. For example, (Gashteovski et al., 2021) discovered that a simple OpenIE system that breaks the sentence into a triple at the verb boundary achieves 0.70 recall and 0.19 precision. This is problematic since it indicates that simply adding extraneous words to the extraction results in improved recall.

BenchIE benchmark. Due to the issues identified for CaRB, we also evaluate using BenchIE, which is an exhaustive fact based multilingual OpenIE benchmark proposed by (Gashteovski et al., 2021). BenchIE evaluates explicit binary extractions in English, Chinese and German. BenchIE is accompanied by an annotation tool, AnnIE (Friedrich et al., 2021), for extending the benchmark to additional languages. We translated 100 sentences from the BenchIE-English to Arabic with the help of a native Arabic speaker and then extracted triples using AnnIE. This allows us to also evaluate our system on Arabic.

Multilingual CaRB. Additionally we also evaluate MILIE on the Spanish and Portuguese mul-

	Chinese			German			Arabic			Spanish (LM)			Portuguese (LM)			Spanish-Clean (LM)		
	F1	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**
M2OIE	17.1	25.7	12.8	4.0	8.9	2.6	4.9	16.3	2.9	60.2	59.1	**61.2**	59.1	56.1	**62.5**	53.5	66.0	44.9
milIE	**20.5**	**25.2**	17.3	8.5	13.4	6.3	—	—	—	**64.2**	69.5	59.7	**65.6**	70.2	61.6	55.7	58.1	53.5
- DEP	19.2	19.8	**18.7**	8.4	11.3	6.7	7.3	**14.2**	4.9	48.1	64.4	38.4	46.9	58.8	39.0	45.0	62.0	35.3
- NS	17.3	19.6	15.5	**10.3**	**14.3**	**8.0**	4.0	10.8	2.5	59.1	**75.7**	48.5	62.4	**74.0**	54.0	**59.5**	**66.2**	**53.9**
- Bin	20.0	22.0	18.4	9.0	13.5	6.7	**7.5**	13.8	**5.1**	—	—	—	—	—	—	—	—	—

Table 3: MILLIE performance comparison on multilingual BenchIE and CARB lexical match (LM) benchmarks. - DEP represents MILIE trained and evaluated without dependency tags, -NS represents absence of negative sampling, -Bin represents lack of binarizing mechanism. MILIE always outperforms M2OIE, except for the recall on the erroneous automatic translation of Spanish and Portuguese. For Arabic no dependency tags were available, therefore the first entry for Arabic is in the line - DEP.

tilingual CaRB datasets introduced in Ro et al. (2020). The lexical match evaluation used in this dataset has numerous shortcomings (Bhardwaj et al., 2019), however we include it for a fair comparison to Ro et al. (2020)'s Multi2OIE system. The CARB test set was translated to Spanish and Portuguese using the Google Translate API. To investigate the quality of these automatic translations, we randomly sampled 100 sentences from the test sets and had them evaluated by native Spanish and Portuguese speakers. To our surprise we discovered that around 70 percent of the sentence or extraction translations were inaccurate. Table 2 shows a few examples of the incorrect translations. For an accurate and clean comparison with Multi2OIE we also cleaned up part of the Spanish test set by re-translating 149 sentences and their extractions in Spanish. These translations were done by native Spanish speakers.

On the CARB English benchmark we use results for baselines reported in (Ro et al., 2020) and (Kolluru et al., 2020a). For evaluating on BenchIE, we run all the baselines on the BenchIE English evaluation benchmark. For multilingual BenchIE we train Multi2OIE using the code and hyperparameters supplied in the paper. For hyperparameter tuning we use the CARB English validation set and use the F1 scores obtained using the CARB evaluation procedure for comparing models with different hyperparameters. The MILIE model is trained using negative sampling and includes the dependency tag information and binarization. We use the spaCy dependency parser for obtaining dependency tags. We were unable to find a dependency parsing tool with universal dependencies for Arabic and therefore we did not use dependency tags for Arabic. For BenchIE, MILIE uses the binarization function described in Section 2.5, but not for CARB and lexical match because they evaluate n-ary extractions.

3.2 Results

3.2.1 English

In Table 4, we compare MILIE with several unsupervised and supervised baselines in English on CARB and BenchIE. MILIE performs much better compared to other neural baselines on BenchIE. This is not the case for the CARB dataset and the reason behind this is that CARB punishes compact extractions while rewarding overly long extractions (Gashteovski et al., 2021). Although rule based systems like ClausIE and MinIE outperform neural systems, they cannot be used for languages other than English.

3.2.2 Multilingual

In Table 3, we compare MILIE with Multi2OIE (M2OIE) on the multilingual BenchIE benchmark. MILIE performs significantly better compared to Multi2OIE for all the languages. For German and Arabic both Multi2OIE and BenchIE perform much worse compared to other languages. The presence of separable prefixes in German verbs which cannot be extracted using BIO tags results in low performance for German. The BIO tagging scheme assumes continuity of phrases which is absent for most German verbs present in predicates, resulting in extremely low recall. For Arabic, the low scores are due to the Verb-Subject-Object nature of the Arabic language along with the fact that subjects or objects can be expressed as part of the verb. This calls for additional research on framing OpenIE tasks for languages such as German and Arabic.

Ablation results in Table 3 also indicate the usefulness of adding the dependency tags, negative sampling and the binarization mechanism. MILIE without negative sampling works best for Spanish clean data. This is not due to the language, but due to the lexical match evaluation which rewards overly long extractions even if incorrect. Not us-

English	CaRB-nary			BenchIE-binary		
	F1	Prec.	Rec.	F1	Prec.	Rec.
ClausIE	44.9	—	—	33.9	50.3	25.6
MinIE	41.9	—	—	33.7	42.9	27.8
Stanford	23.0	—	—	13.0	11.1	15.7
R-OIE	46.7	55.6	40.2	13.0	37.3	7.8
S-OIE	49.4	60.9	41.6	—	—	—
OIE6	**52.7**	—	—	25.4	31.1	21.4
M2OIE	52.3	**60.9**	**45.8**	22.8	39.2	16.1
milIE	45.0	48.6	41.8	**27.9**	**36.6**	**22.4**
-DEP	41.2	44.1	38.6	26.7	31.1	23.4
-NS	44.7	47.6	42.2	25.8	29.6	22.9
-Bin	—	—	—	27.7	34.6	23.1

Table 4: MILIE performance comparison on CARB and BenchIE English benchmarks. MILIE performs best out of all models on BenchIE. It performs worse compared to some model on CaRB, which is due to the CaRB evaluation scheme where overly long extractions are rewarded.

English	F1	Prec.	Rec.	Δ F1
MILIE	27.88	36.65	22.37	—
MILIE + CO	**29.71**	32.35	**27.48**	**+ 6.5 %**

Table 5: Performance comparison of Hybrid MILIE on English BenchIE. Here '+ CO' denotes system fused with extracted ClausIE Objects.

ing negative sampling sometimes improves recall which may improve F1 score. This is observed for the German benchmark.

3.2.3 Hybrid OpenIE

MILIE can easily integrate any rule based system that extracts even a part of the triple. To evaluate this, we first simulate a system that only extracts the object and use MILIE to extract other parts of the triple. We do this by employing ClausIE for extracting triples for the BenchIE English data and only use the object, discarding the rest of the triple.

The reason behind the choice of selecting object extraction from ClausIE is the fact that neural systems are not good at extracting objects (Kolluru et al., 2020a). This is also seen from additional experiments detailed in Section 4. Table 5 indeed confirms that combining rule based object extraction with MILIE improves performance by over 6% in F1 score. This showcases that MILIE's ability to integrate other systems can be a great advantage.

F1-Score	EN	DE	ZH	AR	ES
SPOA	26.3	8.7	20.3	5.3	55.2
SOPA	24.9	8.2	18.2	5.8	53.1
PSOA	27.7	**8.8**	19.5	5.0	51.4
POSA	27.4	8.1	19.4	5.4	51.7
OSPA	22.4	8.0	17.1	5.7	45.5
OPSA	22.2	7.9	17.5	6.4	47.9
WF	**27.9**	8.5	**20.5**	**7.3**	**55.7**

Table 6: Comparison between different decoding schemes. WF represents water filling.

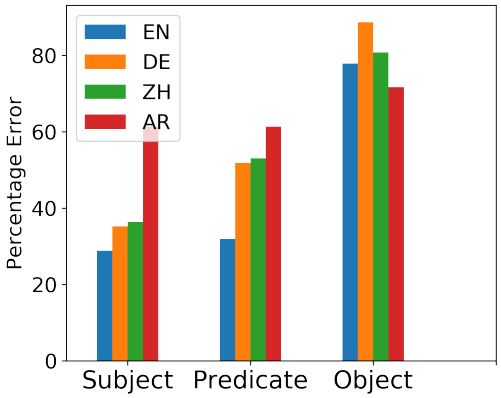

Figure 3: Percentage error contribution due to incorrect subject, predicate or object for EN, DE, ZH and AR. Most errors occur in the object.

4 Analysis

We would like to analyze that the ability of MILIE to extract triples using different extraction patterns results in improved performance on multilingual data. For this, we compare MILIE with the water filling aggregation against MILIE with different extraction pathways.

Table 6 details the performance for different extraction schemes. All the extraction schemes except WF, use only one pathway while WF combines multiple pathways resulting in WF performing much better on all languages, except German. This demonstrates that combining triple extraction from multiple pathways is better than any single pathway, which in turn confirms that extracting triples repeatedly from the same sentence using multiple extraction pathways is more profitable than using a single extraction pathway.

Additionally, Table 6 provides an interesting insight: predicate first seems to be the best, followed by subject first and then object first for languages other than Arabic. This also shows how the difficulty of extracting triple slots using transfer learning from English varies with the target language.

Entropy	Subject		Predicate		Object	
	DEP	POS	DEP	POS	DEP	POS
EN	1.719	1.588	2.443	1.831	2.286	1.861
ZH	2.464	1.827	2.497	1.476	2.602	1.943
DE	1.587	1.567	1.811	1.457	2.115	2.095

Table 7: Entropy of dependency and part of speech tags for subject, predicate and objects in BenchIE test data. Objects exhibit the highest complexity.

Table 6 suggests that predicates are easier to extract leading to lesser number of errors propagated in the prediction chain. We suspect that this could result from differences in linguistic variability. To test our hypothesis we measured the entropy of the distribution of dependency and part-of-speech tags in the predicate, subject and object slots in the BenchIE English and the multilingual test sets. Results shown in Table 7 suggest that linguistic complexity of objects is higher than those of predicates and subjects.

This is also confirmed in Fig. 3, where we plot the extraction errors in either subject, predicate or objects among incorrectly extracted triples. Most errors result from extracting incorrect objects compared to predicates and subjects. The percentage sum does not add to hundred because an incorrect triple can contain errors in more than one slot.

5 Related Work

OpenIE systems largely come in two flavors, (1) unsupervised OpenIE systems that use fine grained rules based on dependency parse trees (Del Corro and Gemulla, 2013; Gashteovski et al., 2017; Lauscher et al., 2019), and (2) supervised neural OpenIE systems, trained end-to-end with large training datasets (Stanovsky et al., 2018; Ro et al., 2020; Kolluru et al., 2020a). Neural OpenIE systems characterize OpenIE as either a sequence tagging task (Stanovsky and Dagan, 2016; Ro et al., 2020), span prediction task or a sequence generation task (Kolluru et al., 2020b). However all these prior approaches extract a triple in a single step, which does not allow us to study the effect of extracting a specific slot and its effect on extracting the rest of the triple.

Neural generative approaches to OpenIE use sequence-to-sequence models with a copy mechanism for generating triples (Sun et al., 2018; Kolluru et al., 2020b). The copy mechanism needs to be learned and is often a source of errors. A series of alternative approaches cast OpenIE as a sequence tagging task where each token is tagged as subject, predicate or object using a BIO like tagging scheme (Stanovsky et al., 2018; Ro et al., 2020; Kolluru et al., 2020a). In these systems, all triple slots are extracted simultaneously and it is therefore not possible to condition on easier slots.

More closely related to our work is SpanOIE (Zhan and Zhao, 2020) and Multi2OIE (Ro et al., 2020), which first extracts the predicate and then all additional arguments. Like us, Multi2OIE (Ro et al., 2020) addresses multilinguality by leveraging a pretrained BERT model (Devlin et al., 2019) for transfer learning. In contrast, through our iterative nature, it is possible to enrich the extractions in other languages if rule based models or other models (e.g. NER recognizers) exist to provide input for a triple slot. IMOJIE (Kolluru et al., 2020b) iteratively extracts entire triples from a sentence: first a triple is extracted, which is added to the input to extract the next triple. In contrast, our work iteratively extracts the slots of a single triple, which allows us to condition on the easier slots and therefore obtain higher quality triples. (Kolluru et al., 2020a) propose OpenIE6, a Bert based system, with iterative grid labelling and linguistic constraint based training. Such lingusitic constraints with soft penalties cannot be readily ported to other languages since such constraints use head verb based heuristics. Consequently OIE 6 is evaluated only on English.

6 Conclusion

We introduced MILIE, a modular & iterative multilingual OpenIE system. We confirmed our hypothesis that it is beneficial to extract triple slots iteratively which allows us to extract easier slots first. Our experiments on English as well as five low resource languages uncovered that, with the exception of Arabic, triples are easier to extract if the predicate is extracted first followed by the subject and object. More importantly we discovered that extracting triples using multiple extraction pathways is superior than the standard single extractions especially in the multilingual setting. We also demonstrated how MILIE can be combined seamlessly with rule based systems for improving performance. Although our experiments were focused on the OpenIE task, we believe that the insights gained can be translated to other information extraction tasks with coupled extractions. We plan to explore such connections in the future.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging Linguistic Structure For Open Domain Information Extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 344–354.

Sangnie Bhardwaj, Samarth Aggarwal, and Mausam Mausam. 2019. CaRB: A Crowdsourced Benchmark for Open IE. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6263–6268.

Luciano Del Corro and Rainer Gemulla. 2013. ClausIE: Clause-Based Open Information Extraction. In *Proceedings of the International World Wide Web Conferences (WWW)*, pages 355–366.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186.

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.

Niklas Friedrich, Kiril Gashteovski, Mingying Yu, Bhushan Kotnis, Carolin Lawrence, Mathias Niepert, and Goran Glavaš. 2021. Annie: An annotation platform for constructing complete open information extraction benchmark. *arXiv preprint arXiv:2109.07464*.

Kiril Gashteovski, Rainer Gemulla, and Luciano Del Corro. 2017. MinIE: Minimizing Facts in Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2630–2640.

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*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 311–316, Vancouver, Canada. Association for Computational Linguistics.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*.

Keshav Kolluru, Vaibhav Adlakha, Samarth Aggarwal, and Soumen Chakrabarti. 2020a. OpenIE6: Iterative Grid Labeling and Coordination Analysis for Open Information Extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3748–3761.

Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Mausam, and Soumen Chakrabarti. 2020b. IMoJIE: Iterative Memory-Based Joint Open Information Extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5871–5886.

Anurag Kumar, D Manjunath, and Joy Kuri. 2008. *Wireless networking*. Elsevier.

Anne Lauscher, Yide Song, and Kiril Gashteovski. 2019. MinScIE: Citation-centered Open Information Extraction. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 386–387. IEEE.

Mausam Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 4074–4077. AAAI Press.

Madhav Nimishakavi, Uday Singh Saini, and Partha Talukdar. 2016. Relation schema induction using tensor factorization with side information. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 414–423, Austin, Texas. Association for Computational Linguistics.

Youngbin Ro, Yukyung Lee, and Pilsung Kang. 2020. Multi^2OIE: Multilingual open information extraction based on multi-head attention with BERT. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1107–1117, Online. Association for Computational Linguistics.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a Large Benchmark for Open Information Extraction. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2300–2305.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised Open Information Extraction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 885–895.

Mingming Sun, Xu Li, Xin Wang, M. Fan, Y. Feng, and P. Li. 2018. Logician: A unified end-to-end neural approach for open-domain information extraction. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

Yumo Xu and Mirella Lapata. 2021. Generating Query Focused Summaries from Query-Free resources. In *Proceedings of the 59th Annual Meeting of the*

Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL), pages 6096–6109.

Zhao Yan, Duyu Tang, Nan Duan, Shujie Liu, Wendi Wang, Daxin Jiang, Ming Zhou, and Zhoujun Li. 2018. Assertion-based qa with question-aware open information extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Junlang Zhan and Hai Zhao. 2020. Span Model for Open Information Extraction on Accurate Corpus. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 9523–9530.

A Appendix

A.1 Training Details

MILIE is expected to predict slots iteratively conditioned on prior extracted slots of a triple, therefore it needs to be trained with similar examples. Exhaustively listing all possible combinations of prior extracted slots and slots to be extracted is prohibitively expensive. Therefore we use a sampling procedure that ensures the model sees a variety of combinations during training.

For every example in the Re-2016 training dataset we do the following

1. Sample an slot as target (for extraction) with the following probabilities (subject: 1/3, predicate: 5/12, object: 5/12)

2. Sample two slots, one that is assumed to be extracted and other the target that needs to be extracted conditioned on the first.

3. Sample three slots, first two assumed to be extracted and the third is the target conditioned on first two.

4. If the example contains n-ary arguments, the subject, predicate and object are assumed to be extracted and the n-ary arguments are treated as targets.

When a slot is sampled for target extraction, all instances of the slot are expected to be extracted. For example, if the target is the subject and if the example consists of multiple subjects then the targets are multiple subjects. However the sampled slots assumed to be extracted must be single instances, and if there are multiple instances, then each instance is considered for conditioning one after the other. Table 8 details the sampling probabilities for two and three slots. The sampling probabilities were not tuned, but rather chosen based on heuristics. Post sampling, we obtain training dataset with about 5 and a half million examples.

Extracted Slots	Target Slot	Probability
subject	object	3/12
subject	predicate	1/12
object	subject	2/12
object	predicate	1/12
predicate	subject	2/12
predicate	object	3/12
(subject, object)	predicate	2/12
(subject, predicate)	object	6/12
(object, predicate)	subject	4/12

Table 8: Sampling Probabilities for training data.

Negative Sampling

We provide MILIE with negative samples during training for reducing error amplification arising out of iterative prediction. In this case the target is always blank, i.e., all the tokens are marked as 'outside'. Thus the sampling revolves around creating incorrectly extracted slots. We sample negatives for every example in the training data and then select k negative samples uniformly at random. k is treated as a hyperparameter.

Table 9 provides the sampling probabilities for different slot arrangements. We use three corruption procedure for generating incorrectly marked slots, namely, invert, randomize and switch. The invert method consist of swapping the extracted slot with the target slot. For example, if the extracted slot is subject and target slot is object, then the object is marked as subject. The randomize method consists of choosing a random span of tokens near the actual slot. Finally the switch method involves switching one of the extracted slot with a slot from another triple associated with the sentence. For example, in the case of (subject, object), the object of this triple is switched with an object of another triple associated with the same sentence. It is possible that the same subject maybe associated with the new object as well. We check if this is true, and if true we filter out such positives.

A.2 Hyperparameter Tuning

We train and evaluate MILIE on an NVIDIA Titan RTX with 24 GB GPU RAM. The training is done for a maximum of two epochs and each epoch takes

Extracted Slots	Target	Corruption	Prob.
subject	object	Invert	1/12
object	predicate	Invert	3/12
predicate	subject	Randomize	2/12
(subject, object)	predicate	Switch	1/12
(subject, predicate)	object	Switch	3/12
(predicate, object)	subject	Switch	2/12

Table 9: Negative Sampling Probabilities.

Num. NS (k)	Learning Rate	F1
0	1×10^{-5}	39.88
0	3×10^{-5}	44.70
0	9×10^{-5}	40.76
10K	1×10^{-5}	43.45
10K	3×10^{-5}	47.03
10K	9×10^{-5}	47.19
100K	1×10^{-5}	48.03
100K	3×10^{-5}	47.30
100K	9×10^{-5}	45.87
1M	1×10^{-5}	46.01
1M	3×10^{-5}	46.16
1M	9×10^{-5}	45.26

Table 10: Hyperparameter Tuning

about 9-10 hours. The maximum sentence length using the English train and validation dataset is found to be about 100. Due to the addition of extracted triple element markers we allow a slack of 20 tokens, thus fixing the maximum sentence length to 120. We use a maximum possible batch size that fits inside the GPU, which results in batch size of 192. We use ADAM (Kingma and Ba, 2015) as the optimizer with linear warmup and tune the learning rate. The linear warmup fraction is fixed at 0.1. We also treat the number of negative samples, k, as a hyperparameter and tune it. We choose the best hyperparameters based on the recall score as it is a better metric for deciding the ability of a model to aggregate triples from different extraction pathways. Table 10 provides details on the recall scores for every hyperparameter arrangement.

A.3 Additional Results

We also compared with a dynamic decoding scheme where MILIE chooses a decoding pathways based on the sentence. To do this we split a part of the English training set and for each sentence in the split we record the extraction pathway that provides the best F1 score MILIE as per CARB evaluation.

F1-Score	EN	DE	ZH	AR	ES
SPOA	26.27	8.65	20.32	5.30	55.21
SOPA	24.95	8.19	18.24	5.80	53.10
PSOA	27.70	8.75	19.46	5.00	51.41
POSA	27.42	8.12	19.43	5.40	51.71
OSPA	22.35	7.96	17.09	5.73	45.45
OPSA	22.24	7.93	17.47	6.40	47.91
DYN	26.90	**8.95**	19.46	4.9	51.00
WF	**27.88**	8.54	**20.50**	**7.30**	**55.70**

Table 11: Comparison between different decoding schemes. DYN represents the dynamic setting.

We then use this as training data for training another mBERT model which classifies each sentence in one of the six classes where each class represents an extraction pathway. While creating the training split we found that for most sentences the extraction path $P_{p,s,o,a}$ results in the best F1 score which unsurprisingly also translates during test time.This is shown in Table. 11

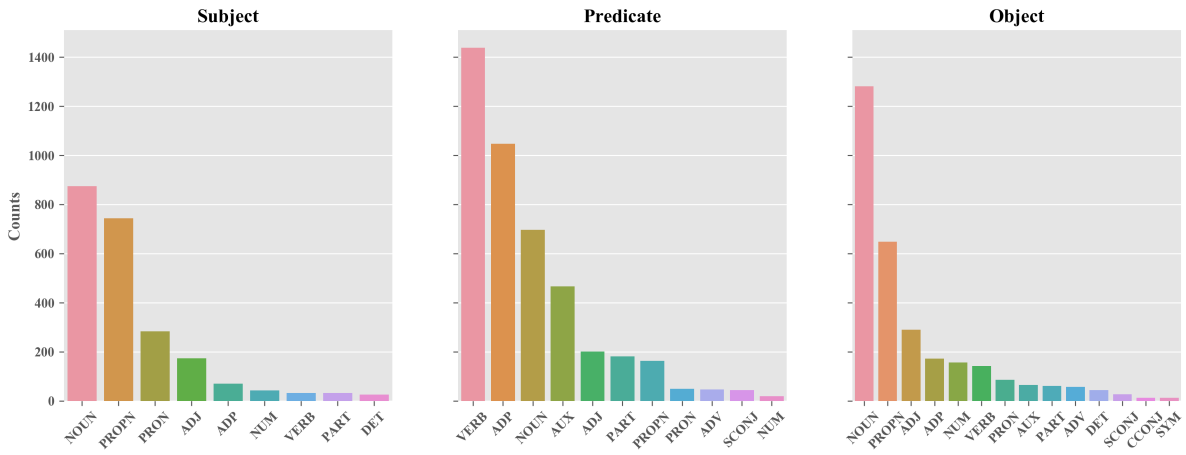

Figure 4: Distribution of the Part of Speech tags in subject, predicates and object tokens of triples in BenchIE English test data.