Language Model Priming for Cross-Lingual Event Extraction

Steven Fincke, Shantanu Agarwal, Scott Miller, Elizabeth Boschee

University of Southern California Information Sciences Institute
sfincke@isi.edu, shantanu@isi.edu, smiller@isi.edu, boschee@isi.edu

Abstract

We present a novel, language-agnostic approach to "priming" language models for the task of event extraction, providing particularly effective performance in low-resource and zero-shot cross-lingual settings. With priming, we augment the input to the transformer stack's language model differently depending on the question(s) being asked of the model at runtime. For instance, if the model is being asked to identify arguments for the trigger *protested*, we will provide that trigger as part of the input to the language model, allowing it to produce different representations for candidate arguments than when it is asked about arguments for the trigger *arrest* elsewhere in the same sentence. We show that by enabling the language model to better compensate for the deficits of sparse and noisy training data, our approach improves both trigger and argument detection and classification significantly over the state of the art in a zero-shot cross-lingual setting.

Introduction

Recent advances in massively-pretrained cross-lingual language models, e.g. Conneau et al. (2020), have revolutionized approaches to extracting information from a much broader set of languages than was previously possible. For some information extraction (IE) tasks, it is not unusual to find annotated datasets in a variety of languages. Name annotation, for instance, has both wide utility and can be performed relatively cheaply and easily by non-experts—and as a result, one can find annotated named entity datasets for languages from Spanish to Polish to Farsi to Indonesian. In contrast, datasets for event extraction tasks are much fewer and further between (even in English).

In order to extract events from most languages, then, it is critical to be able to train models with data from one language and deploy those models in another. A pretrained, cross-lingual language model can carry some of this burden. If the language model's vector representation for *arrest* (English) is close to its representation for *anholdelsen* (Danish), then a model trained on the English sentence "*They protested his arrest*" may be able to detect the ARREST-JAIL event in "*De tre mænd blev ved* ***anholdelsen*** *29 januar*"[1].

[1]The three men were arrested on January 29. (Danish)

One major advance in recent years was the move from static word embeddings, e.g. word2vec (Mikolov et al. 2013), where the representation for a word is constant no matter the context in which it appears, to contextualized language models, e.g. ELMo (Peters et al. 2018) and BERT (Devlin et al. 2019), where the representation for a word is conditioned on its surrounding context. With contextualized language models, the representation for *arrest* in the context "*They protested his arrest*" will likely be very different than when it appears in "*She is a cardiac arrest survivor*".

However, even with a powerful mechanism like BERT, the representation for, say, *activists*, in "*Activists protested his arrest*" will be the same whether the model is being asked if *activists* is an argument of the DEMONSTRATE event ***protested*** or of the ARREST-JAIL event ***arrest***.

Recent advances for structured NLP tasks, such as named entity recognition (Li et al. 2019a), relation extraction (Li et al. 2019b) and coreference resolution (Wu et al. 2020), have noted this and responded accordingly, by providing a *prompt* that modifies the context in which a sentence is seen, allowing the language model to adjust its representations of the sentence tokens accordingly. This method for injecting task-specific guidance to the model is particularly beneficial in low-resource settings (Scao and Rush 2021).

Returning to the task of event extraction, Du and Cardie (2020) show some of the promise of applying similar principles in this domain, reframing event extraction as a question-answering problem. Their best results come from using natural English questions as prompts to their overall model, e.g. asking "*Which is the agent in arrested? — The police arrested the thief.*" However, it is not clear that this English-centric approach of question generation will generalize well to cross-lingual transfer.

In this work, we present a new, language-agnostic mechanism IE-PRIME which provides task-specific direction to an event extraction model at runtime. We show that it significantly outperforms prior work on argument extraction, including Du and Cardie (2020), and that it is particularly effective in low-resource and zero-shot cross-lingual settings for both trigger and argument detection and classification.

Related Work

Event extraction is a well-studied topic. Some of the most effective recent approaches in a mono-lingual context in-

clude Wadden et al. (2019) (DyGIE++), which combines local and global context using contextualized span representations across three unified subtasks (name, relation, and event extraction), and Lin et al. (2020) (OneIE), which uses a joint neural framework to extract globally optimal IE results as graphs from input sentences. Our specific approach draws inspiration from work in question answering. Although several recent works have been likewise inspired, (e.g. Du and Cardie (2020), Feng, Yuan, and Zhang (2020), (Liu et al. 2020)), many of these approaches use natural-language questions and are not therefore an ideal fit for cross-lingual applications. A major drawback of these other QA inspired systems is that they are designed and trained to extract only single answer spans and can only extract multiple spans at decode time by making selections based on a probability cutoff. We make an important improvement over these previous works by avoiding the canonical QA architecture and using a sequence-labelling approach instead, which is inherently capable of finding multiple "answers" from a given question context, e.g. simultaneously identifying more than one VICTIM of an ATTACK event, and avoids any discrepancy between training and decoding behavior. We compare against relevant prior work in both monolingual and cross-lingual contexts and show improved performance for argument extraction in all conditions and improved performance for trigger extraction in the cross-lingual context.

The primary focus of this work is on cross-lingual event extraction. However, most recent cross-lingual event extraction work (e.g. Subburathinam et al. (2019), Ahmad, Peng, and Chang (2021), and Nguyen and Nguyen (2021)) focuses solely on the event argument role labeling task (ignoring trigger detection) and requires pre-generated entity mentions to serve as candidate arguments. Moreover, all of these approaches rely on structural features in both languages, including but not limited to dependency parses. In contrast, our approach does not require any extraction of linguistic structure in the target language. Our argument extraction approach also addresses the problem of candidate identification in addition to labeling, which is critical in real-world applications. We report results both on the limited task performed by the above-mentioned papers, showing improvements over the reported state of the art, as well as on the complete end-to-end task.

Approach

In this section we describe our baseline system (IE-BASELINE) and an extension of that system (IE-PRIME) that provides significant improvement. Both systems are designed to be completely language-agnostic—documents in any language supported by the language model can be used as either training or test data—and as such can be applied in either a monolingual or cross-lingual context.

IE-BASELINE

Our baseline system consists of two trained components, one for event trigger extraction and one for argument attachment. Trigger detection and classification are performed using a simple beginning-inside-outside (BIO) sequence-labeling architecture composed of a single linear classification layer on top of the transformer stack.

Our baseline argument extraction system (shown in Figure 1) takes as its input a proposed event trigger and from an input sentence identifies argument spans and labels them with argument roles. This system produces arguments by generating BIO argument labels over the sentence tokens using a bi-LSTM, feeding into a single linear layer and then to a CRF-based loss function[2] atop a transformer-based language model. The input for each token is the concatenation of the transformer output for the token itself, along with the transformer output for the trigger token, and an embedding for the event type; in our default configuration, no input entity information is used. When an individual input token is split up by the language model's tokenizer (e.g. if *characteristically* is split into *characteristic* and *##ally*), the average of the output vectors for the parts represents the whole.[3] All models fine tune all the layers of the language model and only use the output from the final layer.

IE-PRIME

The key to our proposed system, IE-PRIME, is "priming": the idea that we can augment the input to the transformer stack in a way that provides critical additional information to the model at runtime.

We first describe how we do this for the task of argument extraction. Our baseline argument extraction system is designed to take as input a single trigger and a sentence. It then produces a complete set of argument spans and roles with respect to that trigger. It repeats this process for every proposed event trigger.

The first form of priming we explore leverages that trigger. Specifically, we augment the input to the language model by pre-pending the trigger to the sentence being considered (divided from the sentence by a sentence-separating token appropriate to the language model being used, e.g. `[SEP]` for BERT). So, if a model trained using BERT is seeking arguments of *protested*, the language model would receive the following input: `[CLS] protested [SEP] crowds protested the conviction of Ildar Dadin [SEP].` However, when searching for arguments of *conviction*, the following input is used instead: `[CLS] conviction [SEP] crowds protested the conviction of Ildar Dadin [SEP].` Figure 2 shows a graphical representation of the second example.

We note that our baseline system already indicates the trigger of interest when predicting arguments (by appending its vector to the vector representation for each token), so priming is arguably redundant. However, providing this input in a second way appears to enable the language model to respond more effectively to the change in focus at runtime. This redundancy also allows the system to handle the case where an event trigger word occurs more than once in a sentence e.g. *The company **fired** John a week after they **fired***

[2] As implemented in torchcrf, https://pytorch-crf.readthedocs.io/en/stable/

[3] We also explored other strategies, e.g. taking only the first vector, but averaging worked best.

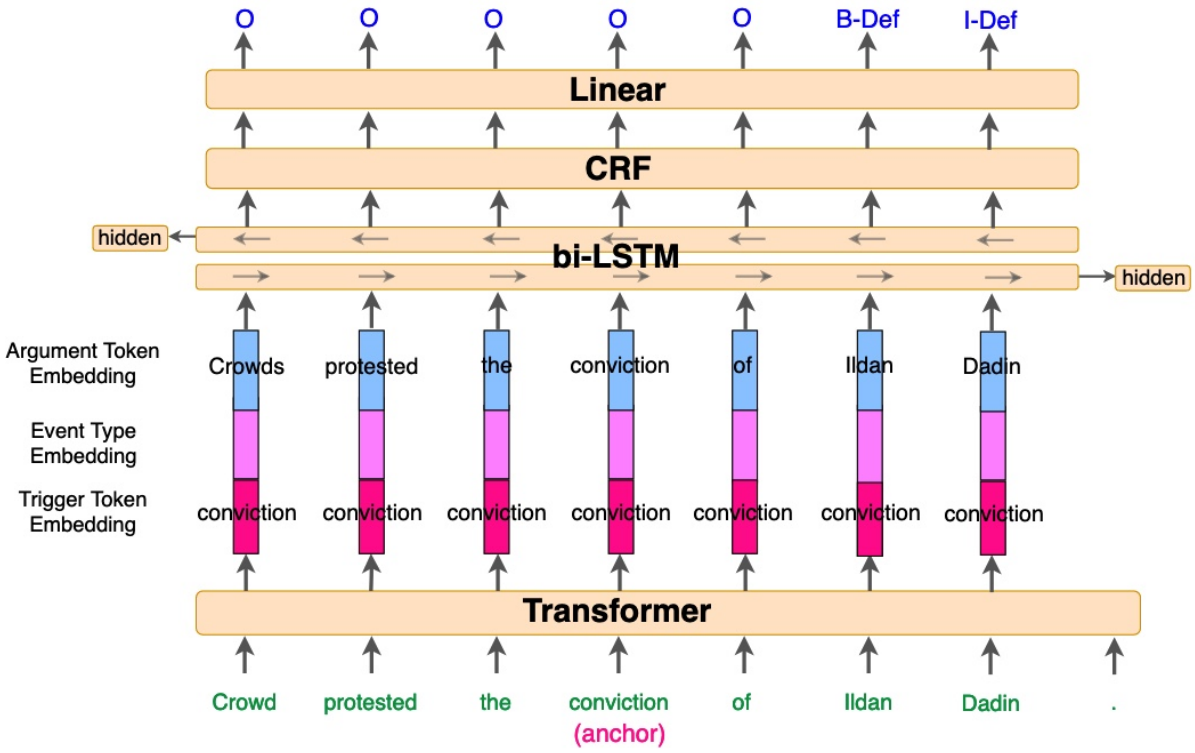

Figure 1: Baseline argument attachment architecture.

Bob. The placement of the trigger *fired* at the beginning of the sentence would not be enough to disambiguate between the two instances of *fired* in the sentence, but because the vector representation for the trigger token is generated by a contextualized language model from a specific token in the sentence, the model can distinguish between the *fired* that should have *John* as an argument and the *fired* that should have *Bob*.

The second form of priming we explore leverages both the trigger and the argument role.[4] Here, rather than asking the model for all arguments of a trigger at the same time, we query separately for each possible type of argument. That is, we query first for any DEFENDANT arguments of a CONVICT event, and then separately for any ADJUDICATOR arguments, and so on. In this formulation, we augment the input to the language model with both the trigger and the argument role. However, to facilitate application to a variety of languages, we replace each English argument role label with a unique integer string, giving a result like this when asking about any DEFENDANT arguments of *conviction*: `[CLS] conviction ; 13 [SEP] crowds protested the conviction of Ildar Dadin [SEP]`. Figure 3 shows a graphical representation.

Both forms of priming described above can be used to either generate BIO labels or label pre-identified candidate arguments. We found priming by trigger and argument role to be generally more successful and is what we report below unless otherwise noted.

We note that when priming by trigger and argument role together, it is possible for the model to predict multiple roles for a single span. This is consistent with the definition of the ACE task, where there is no requirement that arguments with different roles not overlap, and in fact sometimes they do (e.g. the same text span plays more than one role in an event). For other event tasks where this might *not* be allowed, priming by the trigger alone would be the most straightforward path.

Finally, we also developed a priming model for trigger detection and classification. Here, the input is primed with a single token from the sentence. So, the language model again sees something like this: `[CLS] conviction [SEP] crowds protested the conviction of Ildar Dadin [SEP]`. However, the question asked of the model is not about the arguments of *conviction* but about whether *conviction* is itself part of a trigger, and if so, what kind. This model targets two training objectives. One objective predicts the output span for the trigger with BIO labelling with a bi-LSTM and CRF, similar to argument extraction. The other objective takes the concatenation of the language model output for the trigger token and the class token as input and applies a linear transformation to predict the event type.

Thus, in the example above, we expect *conviction* to be marked as the trigger span and the lae nguage model outputs from `[CLS]` and *conviction* to lead the model to generate

[4]Different argument roles are possible for different ACE event types: our system only queries the roles allowed for the event type specified for the trigger.

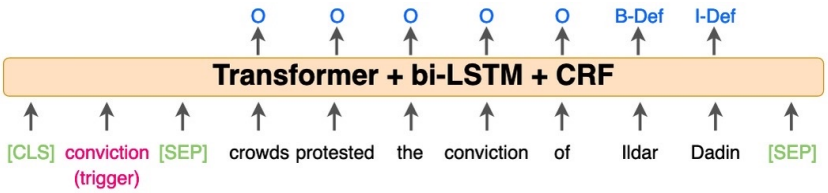

Figure 2: Priming a sentence for the trigger *conviction*. The span *Ildar Dadin* is identified as a DEFENDANT argument.

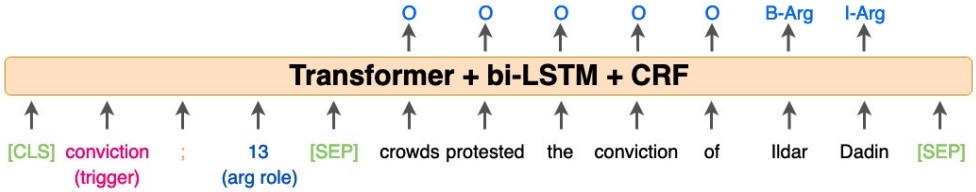

Figure 3: Priming a sentence for the trigger *conviction* and the argument role DEFENDANT. The span *Ildar Dadin* is identified as an argument and is therefore assigned the role being queried (DEFENDANT).

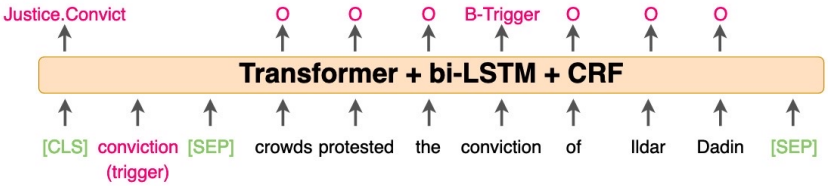

Figure 4: Priming a sentence to determine whether *conviction* is part of a trigger span. The span *conviction* is identified as a trigger of the type JUSTICE.CONVICT.

JUSTICE.CONVICT as the event type; Figure 4 shows this in graphical form. The system outputs a trigger only if and only if an event type is predicted; if BIO labelling provides no span overlapping with the priming token, the model outputs a trigger with just the priming token as its span. When the output for the trigger span and event type conflict, our system favors event type prediction. If no event type is predicted, no trigger will be output, even when a trigger span is predicted. Likewise, if an event type is predicted, a trigger will always be generated; if no predicted trigger span overlaps with the primed token, span prediction is ignored, and the output span is only the primed token.

Variant: Pre-generated Candidate Arguments

To allow better comparison with prior cross-lingual work (e.g. Subburathinam et al. (2019)), we developed a variant of our system which takes as input pre-generated candidate arguments instead of finding argument spans in the sentence at decode time. In this paper, we use this variant to produce results in an experimental setting that uses gold entity mentions as argument candidates. We adapt our architecture for this setting in three small ways.

First, it no longer makes sense to have the model consider arbitrary spans, so instead of all tokens, we present a sequence of candidate arguments and ask the model to classify each with respect to a specified trigger. (We re-use the same architecture: a single linear layer with a CRF-based loss function on top of the transformer stack.) For each candidate, we take as its representation the transformer output for the candidate's "most representative" token, ideally its linguistic head. For instance, the model will use the vector for *Smith* to represent *Bob Smith*, or the vector for *student* to represent *the medical student*. To select this token for each candidate argument, we take the highest-ranking token in its dependency parse, if available[5]; if not, we simply select an argument's first token. (Using dependency parses is not required and typically provides only a small (1-2 point) gain over always selecting the first token.)

Second, we augment our argument model to consider entity type by including the output vector from an entity type embedding in the set of vectors concatenated for the candidate. This entity type embedding is trained along with the rest of the model.

Third, we constrain our system to produce only "legal" argument/trigger pairs; that is, if no entity of type PERSON ever appears as the PLACE argument to an ARREST-JAIL event in training, neither do we allow it to do so at test time.

We note that this approach could also be applied to system-generated candidate arguments, should training data for them exist (e.g. in the ACE2005 dataset). However, in our cross-lingual experiments, our primary system (which

[5]Generated here using spaCy (https://spacy.io/) in English or the model trained from the Prague Arabic Dependency Treebank (https://github.com/UniversalDependencies/UD_Arabic-PADT) for UDPipe (Straka and Straková 2017) in Arabic.

considers all possible spans as arguments) provided superior results, so we did not pursue this variant further except when required for experiments using gold entity mentions.

Experimental Setup

We report results in **two experimental settings**, both using the ACE 2005 corpus (English and Arabic)[6]. We believe the first setting provides a more accurate and complete picture of the full event extraction task, but we include the second (which evaluates only event-argument role labeling) for a full comparison to prior work.

Our **primary experimental setting** uses the standard English document train/dev/test splits for this dataset (Yang and Mitchell 2016) and the Arabic splits proposed by Xu et al. (2021). In both cases, we draw sentence breaks from the data made available by Xu et al. (2021), which is generated using the DyGIE++ codebase.[7] We further split Arabic sentences at runtime to ensure a maximum length of 80 tokens. (We still score against the unsplit reference.) We also use Farasa (Abdelali et al. 2016) to remove tatweels and map presentation forms to the standard code range. All results reported in this paper other than Table 2 use this experimental setting and are the average of five seeds.

Our **secondary experimental setting** evaluates only event-argument role labeling. It replicates the conditions proposed by Subburathinam et al. (2019), which ask systems to label individual instances consisting of a gold trigger and a gold entity mention. The train/dev/test split does not consider the origin of each instance, meaning that the same sentence and trigger can be found in both train and test, with different candidate arguments. This is inconsequential in a cross-lingual setting but should be taken into consideration if assessing monolingual results. Moreover, following prior work, both training and test down-sample the number of negative instances to match the number of positive instances. For this reason, observed precision is dramatically over-inflated compared to a real-world scenario; reported F-measure should be considered in this light. The only table in this paper using this experimental setting is Table 2.

Unless otherwise specified, for **language models** we use the large, cased version of BERT (Devlin et al. 2019) for the monolingual English condition and the large version of XLM-RoBERTa (Conneau et al. 2020) for cross-lingual or Arabic-only conditions. We use BERT for monolingual English solely to ensure a fair comparison to prior work; XLM-RoBERTa could also be used with similar results.

Following community practice for **evaluation metrics**, e.g. Zhang, Ji, and Sil (2019), we consider a trigger correct if its offsets and event type are correct, and we consider an argument correct if its offsets, event type, and role find a match in the ground truth.

[6]https://www.ldc.upenn.edu/collaborations/past-projects/ace

[7]https://github.com/dwadden/dygiepp

Results

Argument Extraction

We begin by assessing the impact of our new priming architecture directly on argument extraction. To do this, we first present results with gold triggers in our primary experimental setting. Table 1 shows improvements from priming across the board in both monolingual and cross-lingual conditions. We also report English results here from Du and Cardie (2020), also inspired by a question-answering approach, and results from all three conditions using a locally trained version of OneIE (Lin et al. 2020) constrained to gold triggers at test time. IE-PRIME comfortably out-performs both prior baselines.

In order to compare against prior work in a cross-lingual setting, we next present analogous results in our secondary experimental setting.[8] This setting represents the narrower task of event argument role labeling (using gold triggers and gold entity mentions). Table 2 presents our results in this experimental condition, where IE-PRIME shows more than a six-point improvement over the best previously-reported results (Ahmad, Peng, and Chang 2021).

We note that this secondary experimental condition is somewhat artificial, not just in its reliance on gold entity mentions (both spans and entity types) but also in its exclusion of 90% of the negative instances during both training *and* test. To mitigate the significant dataset bias in this condition, we ran a version of our system which took as its output the union of arguments found by models trained with five different seeds, allowing us to force the system to prioritize recall.

We also show results for both priming variants for the secondary experimental condition (priming by trigger only and by trigger and role together). In the primary experimental condition, the trigger+role approach to priming is always better and that is what we typically report in this paper (e.g. in Table 1). However, here, the trigger-only version is superior, perhaps because it allows the full model to see at once how many arguments are being predicted for a given trigger and therefore again allows the model to favor higher recall to better approximate the (somewhat unrealistic) distribution in this version of the data.

End-to-End System

To further situate our system within the state of the art, we must consider an end-to-end system, since the currently best-performing event extraction systems jointly optimize extraction of entity mentions, triggers and arguments. Our primed system (as analyzed so far) focuses solely on improving argument extraction performance, and indeed our simple model's trigger performance lags behind the state of the art, particularly in English. Our system also does not use entity annotation, which is used by both of the prior-work baselines shown here. Still, we see in Table 3 that despite weaker trigger performance and without using any entity

[8]We are not aware of any published full system results for English→Arabic cross-lingual ACE event extraction.

	Mono		Cross
	en→en	ar→ar	en→ar
Du and Cardie (2020)	65.4	–	–
Lin et al. (2020)	69.3	56.3	36.4
Ahmad, Peng, and Chang (2021)	–	–	44.5
IE-BASELINE	63.2	53.3	44.7
IE-PRIME	**72.4**	**67.7**	**50.3**

Table 1: Gains in argument classification F1 score from priming for argument extraction using gold triggers in our primary experimental setting. As for all reported results in this paper, IE-PRIME uses the trigger+role configuration for priming unless otherwise specified. To provide Arabic and cross-lingual baselines, (Lin et al. 2020) and (Ahmad, Peng, and Chang 2021) are re-trained and constrained to gold triggers at test time. Because (Du and Cardie 2020) relies on an English query generation framework, we could not run it in the Arabic or cross-lingual condition.

	Priming method	Recall	Precision	F-Measure
Subburathinam et al. (2019)	–			61.8
Ahmad, Peng, and Chang (2021)	–			68.5
IE-BASELINE	–	67.1	79.6	72.8
IE-PRIME	trigger + role	66.5	82.9	73.8
IE-PRIME	trigger	**67.5**	**83.7**	**74.7**

Table 2: Secondary experimental setting: Argument classification F1 in the zero-shot cross-lingual condition (train on English, test on Arabic) with gold triggers and gold entity mentions, following splits from (Subburathinam et al. 2019).

annotation, our monolingual argument extraction in English comes near that of the state-of-the-art system.[9]

Arabic event extraction performance has not been as widely reported as English, and the Arabic training set is only 40% of the size of the English training set, making it a lower-resource condition. Following Xu et al. (2021), we take as a baseline a DyGIE++ model trained on the Arabic dataset. Results in Arabic (monolingual or cross-lingual) confirm the strength of our argument extraction approach with respect to the state of the art, both in the lower-resource Arabic monolingual condition (+4.5 over DyGIE++) and the zero-shot cross-lingual condition (+8.2 over DyGIE++).

Finally, noting that triggers serve as a point of weakness in our baseline system, we further explore the possibilities of priming by testing our primed trigger model in the cross-lingual context. Although we see no gain in the monolingual conditions, we see a very significant gain in the cross-lingual context (from 42.4 to 51.0 for trigger classification), suggesting that the primary strength of priming is enabling the underlying language model to compensate when the task-specific training data is noisy (e.g. from another language).

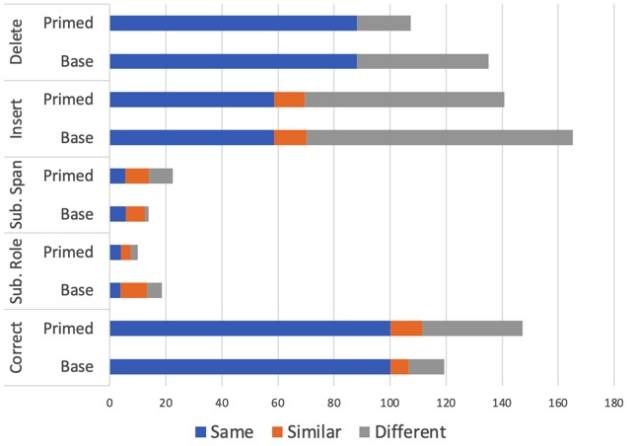

Figure 5: Analysis of differences between IE-BASELINE and IE-PRIME for zero-shot Arabic with gold triggers (averaged over five seeds).

Further Analysis

Error Analysis To focus on the impact of priming, we first analyze differences between the output of IE-BASELINE and IE-PRIME in the zero-shot English→Arabic context (Figure 5). Cases in which a system output span overlaps with with the gold but the exact span or role are incorrect are classified as substitutions; instances when both systems provide outputs which overlap but differ in exact span or role are marked *similar*. Results show that IE-PRIME provides significantly more correct answers overall, adding an additional 47 beyond the 100 found by both systems, while IE-BASELINE adds only an additional 19 not found by IE-PRIME. The total number of substitutions remains constant,

[9]Note that since this is an end-to-end system, trigger performance directly impacts argument extraction performance, as systems are penalized for producing arguments for spurious triggers or missing arguments for missed triggers.

	Monolingal				Cross-lingual	
	en→en		ar→ar		en→ar	
	trigger	argument	trigger	argument	trigger	argument
Wadden et al. (2019)	69.7	48.8	61.5	44.4	41.6	22.0
Du and Cardie (2020)	72.4	53.1	–	–	–	–
Lin et al. (2020)	**74.7**	**56.8**	**64.3**	48.0	49.0	30.8
IE-PRIME (arguments only)	71.2	55.3	61.2	**48.9**	42.4	30.2
IE-PRIME (arguments + triggers)	68.1	52.9	60.2	48.7	**51.0**	**32.4**

Table 3: Trigger and argument classification F1 for end-to-end systems in our primary experimental setting. The first version of IE-PRIME includes the baseline trigger component and the primed argument extraction component. The second version includes both the primed trigger component and the primed argument extraction component. To provide baselines in the Arabic and cross-lingual conditions, (Wadden et al. 2019) and (Lin et al. 2020) are re-trained and run using the large version of XLM-RoBERTa. The baseline numbers reported for the monolingual English condition are taken from the original papers (and as with all monolingual English results reported here, use the large version of BERT). Because (Du and Cardie 2020) relies on an English query generation framework, we could not run it in the Arabic or cross-lingual condition.

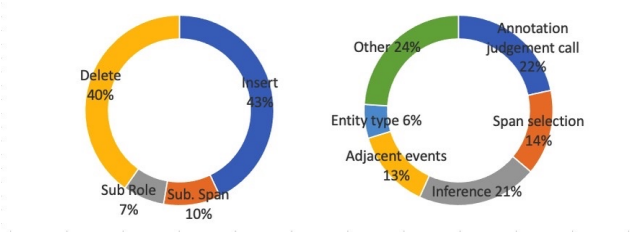

Figure 6: Classification of errors when training and testing on English with gold triggers.

	en→en		ar→ar		en→ar	
	base	large	base	large	base	large
IE-BASELINE	60.0	66.0	46.9	53.3	35.6	44.7
IE-PRIME	69.0	**74.1**	60.8	**67.7**	40.2	**50.3**

Table 4: Comparison of argument classification F1 (using gold triggers) based on size of pretrained language model.

and the increase in correct guesses for IE-PRIME yields a 60% reduction in novel deletions (misses). Comparing deletions to insertions, the two systems are more likely to agree on deletions (items missed by the model), while insertions are much more heterogeneous. This is intuitive: there are more unique ways to hallucinate a false alarm argument than there are unique correct answers to delete. We note that IE-PRIME does reduce novel insertions by a quarter compared to IE-BASELINE, showing that the improvement from priming holds for both recall and precision.

For qualitative analysis, we focus on IE-PRIME in the monolingual English condition.[10] Figure 6 summarizes our observations. Overall, we find the system evenly balanced between deletions (40% of all errors) and insertions (43%). Another 17% are classified as substitutions, with only one of the role (7%) or argument span (10%) being incorrect. We classify observed errors into six categories:

- Annotation judgment call (22%): The system decision seems plausibly correct to a new reader.
- Span selection errors (14%): The model did not select the correct span for an argument, e.g. labelling *Headquarters* when *Security Headquarters* is expected. A higher-level re-scoring of argument spans might be helpful here.
- Inference required (21%): Sometimes the system task requires world knowledge and/or knowledge of document context beyond the local sentence (which our model does not consider). For instance, an earlier sentence might give a clue that a person is a smuggler, making them more likely to be later found as a DEFENDANT in a CONVICT event; incorporating document context is an important next step for our approach.
- Confusion due to adjacent events (13%): Some confusion can arise due to the proximity in the text of other events. For instance, in *Davies is leaving to become chairman of the London School of Economics*, our model correctly marks *London School of Economics* as the *Entity* argument for *become (chairman)*, but we incorrectly also label it as the *Entity* for *leaving*. Adding a mechanism to ensure consistency between events could help here.
- Entity type (6%). A handful of errors involve entities whose semantic class is easy to mistake, e.g. a model likely not understanding *Milton Keynes* as a place name.
- Other (24%): The remaining errors lack any ready explanation, but it appears that some might benefit from more explicit modeling of syntactic information or a re-scoring pass to consider event-to-event interaction.

Pretrained Model Size We believe the gains from priming come from giving the base language model more opportunity to compensate for gaps in the training data. Another common way to improve the performance of an architecture that relies on a pretrained language model is simply to

[10] Only the system using the seed *1235* was examined.

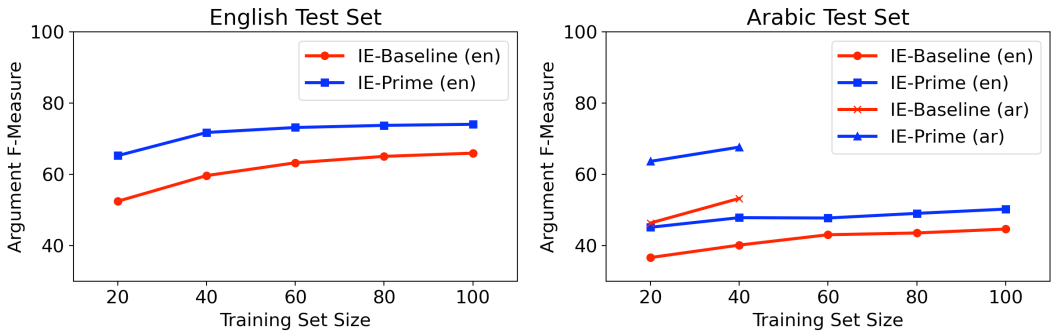

Figure 7: Comparison of IE-BASELINE and IE-PRIME by approximate training set size. Training size here is calculated as the number of events in a document set and is shown as a percentage of full English training set size. The language of the data in which the models were trained are denoted in parenthesis. Experiments in this figure use gold triggers.

increase the size of that model (either the number of parameters users, or the data sets it sees during training, or both). Table 4 presents the results of our baseline and primed models with both the base and large versions of XLM-RoBERTa. As we can see, the gains achieved from priming and from language model size are complementary. In all conditions, we see an improvement over baseline from adding *either* priming or moving to a larger language model, and again in all conditions, we see significant further gains from adding both. We note that the absolute gains from adding priming are quite similar regardless of language model size.

Training Set Size We have hypothesized that priming is particularly effective in low-resource conditions. To test this, we vary the size of our training set and examine the relative gain from priming in each condition. For simplicity, Figure 7 presents results only for argument extraction, with gold triggers. Training size is calculated as the number of events in a document set and shown as a percentage of the English training set size; so, the full Arabic training set (1.7K events) is approximately equivalent to 40% of the English training set (4.2K events).

We make two primary observations. First, the gap between the baseline and primed systems is indeed greatest in the lowest-resource settings, showing that the priming architecture provides more power when the model is under-resourced. For instance, when only 20% of the English data is available (∼850 events), we see a 12.8-point gain from adding priming; when 100% of the data is available, the gain is smaller (8.1 points). The same holds true for the Arabic monolingual and English→Arabic cross-lingual conditions.

Second, we show that priming is able to overcome low-resource conditions rather significantly. In English, IE-PRIME requires only 20% of the training to nearly equal the performance of IE-BASELINE trained with 100% of the data. The same is true in the cross-lingual condition. We also see that in the this lowest-resource condition (20%, or ∼850 events), the performance of IE-PRIME in the cross-lingual condition is actually almost equal to the performance of IE-BASELINE trained on the same amount of native Arabic data. All of these results show the utility of priming when only a small amount of data (or data from the wrong language) is available.

Conclusions & Future Work

We have shown here that our novel priming architecture improves both trigger and argument detection and classification significantly over the state of the art in a zero-shot cross-lingual setting. Our approach also provides significant gains for argument extraction in the monolingual context. However, there are still many areas yet to be explored within this new paradigm. For instance, prior work has shown improvement by jointly modeling triggers and arguments; we expect this would be an area of gain here as well. Prior work has also shown the value of document-level information, none of which is exploited in our current work. We look forward to investigating the incorporation of these elements (and others) as we continue to explore this promising paradigm.

Acknowledgements

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19051600007. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

Many thanks to I-Hung Hsu and Kuan-Hao Huang for retraining and locally re-running the OneIE and GATE systems and providing us with the baseline numbers reported in this paper.

References

Abdelali, A.; Darwish, K.; Durrani, N.; and Mubarak, H. 2016. Farasa: A Fast and Furious Segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, 11–16. San Diego, California: Association for Computational Linguistics.

Ahmad, W. U.; Peng, N.; and Chang, K.-W. 2021. GATE: Graph Attention Transformer Encoder for Cross-lingual Relation and Event Extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; and Stoyanov, V. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8440–8451. Online: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Du, X.; and Cardie, C. 2020. Event Extraction by Answering (Almost) Natural Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Feng, R.; Yuan, J.; and Zhang, C. 2020. Probing and Fine-tuning Reading Comprehension Models for Few-shot Event Extraction. arXiv:2010.11325.

Li, X.; Feng, J.; Meng, Y.; Han, Q.; Wu, F.; and Li, J. 2019a. A Unified MRC Framework for Named Entity Recognition. *arXiv preprint arXiv:1910.11476*.

Li, X.; Yin, F.; Sun, Z.; Li, X.; Yuan, A.; Chai, D.; Zhou, M.; and Li, J. 2019b. Entity-Relation Extraction as Multi-Turn Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1340–1350. Florence, Italy: Association for Computational Linguistics.

Lin, Y.; Ji, H.; Huang, F.; and Wu, L. 2020. A Joint Neural Model for Information Extraction with Global Features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7999–8009. Online: Association for Computational Linguistics.

Liu, J.; Chen, Y.; Liu, K.; Bi, W.; and Liu, X. 2020. Event Extraction as Machine Reading Comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1641–1651. Online: Association for Computational Linguistics.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Nguyen, M. V.; and Nguyen, T. H. 2021. Improving Cross-Lingual Transfer for Event Argument Extraction with Language-Universal Sentence Structures. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, 237–243. Kyiv, Ukraine (Virtual): Association for Computational Linguistics.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237. New Orleans, Louisiana: Association for Computational Linguistics.

Scao, T. L.; and Rush, A. M. 2021. How Many Data Points is a Prompt Worth? *CoRR*, abs/2103.08493.

Straka, M.; and Straková, J. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 88–99. Vancouver, Canada: Association for Computational Linguistics.

Subburathinam, A.; Lu, D.; Ji, H.; May, J.; Chang, S.-F.; Sil, A.; and Voss, C. 2019. Cross-lingual Structure Transfer for Relation and Event Extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 313–325. Hong Kong, China: Association for Computational Linguistics.

Wadden, D.; Wennberg, U.; Luan, Y.; and Hajishirzi, H. 2019. Entity, Relation, and Event Extraction with Contextualized Span Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 5784–5789. Hong Kong, China: Association for Computational Linguistics.

Wu, W.; Wang, F.; Yuan, A.; Wu, F.; and Li, J. 2020. CorefQA: Coreference Resolution as Query-based Span Prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6953–6963. Online: Association for Computational Linguistics.

Xu, H.; Ebner, S.; Yarmohammadi, M.; White, A. S.; Van Durme, B.; and Murray, K. 2021. Gradual Fine-Tuning for Low-Resource Domain Adaptation. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, 214–221. Kyiv, Ukraine: Association for Computational Linguistics.

Yang, B.; and Mitchell, T. M. 2016. Joint Extraction of Events and Entities within a Document Context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 289–299. San Diego, California: Association for Computational Linguistics.

Zhang, T.; Ji, H.; and Sil, A. 2019. Joint Entity and Event Extraction with Generative Adversarial Imitation Learning. *Data Intelligence*, 1: 99–120.