STEPS: Semantic Typing of Event Processes with a Sequence-to-Sequence Approach

Sveva Pepe **Edoardo Barba** **Rexhina Blloshmi** **Roberto Navigli**

Sapienza NLP Group
Sapienza University of Rome
pepe.1743997@studenti.uniroma1.it
{barba,blloshmi}@di.uniroma1.it
navigli@diag.uniroma1.it

Abstract

Enabling computers to comprehend the intent of human actions by processing language is one of the fundamental goals of Natural Language Understanding. An emerging task in this context is that of free-form *event process typing*, which aims at understanding the overall goal of a protagonist in terms of an *action* and an *object*, given a sequence of events. This task was initially treated as a *learning-to-rank* problem by exploiting the similarity between processes and action/object textual definitions. However, this approach appears to be overly complex, binds the output types to a fixed inventory for possible word definitions and, moreover, leaves space for further enhancements as regards performance. In this paper, we advance the field by reformulating the free-form *event process typing* task as a sequence generation problem and put forward STEPS, an end-to-end approach for producing user intent in terms of actions and objects only, dispensing with the need for their definitions. In addition to this, we eliminate several dataset constraints set by previous works, while at the same time significantly outperforming them. We release the data and software at https://github.com/SapienzaNLP/steps.

1 Introduction

Event understanding is a core process of the human perceptual system, which, while conceiving event processes, makes continuous predictions about what will happen next. In fact, cognitive studies state that the perception of events is highly correlated with the prior knowledge about the event parts and the inference of performer's intent and plans (Zacks et al. 2007). In the context of Natural Language Processing (NLP), the Multi-axis Event Process Typing task aims at automatically understanding the overall goal of a performer relying on a set of systematically connected events (Chen et al. 2020). More formally, given a sequence of events, i.e., partially ordered events that are centered around common protagonists (Chambers and Jurafsky 2008), *event process typing* is defined as a free-form labeling task along two axes: the type of *action* the event process seeks to complete, and the type of *object* which is affected by the action of the performer. For instance, if a performer takes the following steps, 1. *boil the water*, 2. *add salt*, 3. *add pasta*, 4. *leave for 12 minutes*, 5. *drain the water*, 6. *add sauce as per preference*, the overall intent is that of COOKING PASTA, where COOK is the action and PASTA is the object. While this might be easy to deduce for humans, it is a challenging task for machines as, for instance, the action and object types may not appear in the event sequence. Event understanding, and especially intent inference, can potentially benefit several downstream applications of Natural Language Understanding (NLU), from commonsense reasoning to dialogue systems. Indeed, successful NLP applications that rely on event understanding include story comprehension for narrative prediction (Chaturvedi, Peng, and Roth 2017), machine reading comprehension for information extraction in biological processes (Berant et al. 2014), and more recently video segmentation (Zhukov et al. 2019; Fried et al. 2020). However, despite the wide range of NLP applications benefiting from event processes understanding, the recent task of *event process typing* is still understudied, and the currently existing approach to it (Chen et al. 2020) suffers from three main shortcomings: i) it relies on indirect supervision from action/object glosses[1] to rank their types based on the similarity with input events, thus requiring a third-party Word Sense Disambiguation (WSD) system to assign a gloss to each action and object, ii) it requires a fixed sense inventory to define the possible action and object types, a binding not in line with the free-form nature of the classes, and finally, iii) it breaks the procedural flow of events by filtering out different steps due to *harsh* dataset filtering, thus harming their sequential nature. With the aim of overcoming these problems, this paper makes the following contributions:

1. we reformulate the Multi-axis Event Process Typing task as a sequence generation problem which maintains and benefits from the free-form nature of the types;
2. we present STEPS, a sequence-to-sequence model that accepts a sequence of events and produces the overall human intent in terms of actions and objects only, thus eliminating the need for their glosses;
3. we do away with various constraints set on the existing dataset and provide an extensive study about the effect these constraints have on the overall performance of the models;
4. we advance the state of the art by a large margin, even when switching in low-data and few-shot settings.

[1] We use the terms *gloss* and *definition* interchangeably.

2 Related Work

Researchers have tried to understand events from a variety of different angles. Previous work has attempted to define several formalisms for inferring protagonists, roles, and other attributes related to events (Baker, Fillmore, and Lowe 1998; Kipper Schuler, Korhonen, and Brown 2009; Das et al. 2014). In addition, particular interest has also been attracted by the event understanding and prediction tasks towards i) event completion (Chambers and Jurafsky 2008; Radinsky and Horvitz 2013), which seeks to predict the missing events in a sequence, ii) membership prediction (Zhang et al. 2020), which produces the whole sequence of events, iii) salience prediction, which determines the importance of each event in a process, and finally, iv) intention prediction (Rashkin et al. 2018; Chen et al. 2020), which is defined as the task of understanding the overall goal of event processes. This latter direction is the most closely related to how we try to understand events in our work. In pursuing this direction, whereas Rashkin et al. (2018) rely on a *sequence generation* approach to produce the most likely intents and reactions of a participant, given a single-cause free-form event description, Chen et al. (2020), instead, focus on processes made up of multiple event descriptions. Indeed, Chen et al. (2020) present the Multi-axis Event Process Typing task and put forward a large dataset in the English language comprising of ultra fine-grained typing instances on two axes, i.e., the action and object types, with vast label vocabularies. To support the introduced task, they propose a *learning-to-rank* approach, which assigns a sorted list of action and object types to a given sequence of events. This ranking is performed by leveraging dense representations of both the input events and a set of action and object descriptions coming from a predefined meaning inventory.[2] Differently from Chen et al. (2020), in this work, we reformulate the Multi-axis Event Process Typing task as a sequence-to-sequence problem, where a system, starting from the list of events that constitute a process, has to generate a sequence of tokens from which we can easily extract both the predicted action and object types. With a view to achieving this, we propose STEPS, which, differently from the *learning-to-rank* approach of Chen et al. (2020), is based on an encoder-decoder model to autoregressively produce the required sequence of tokens. Compared to the sequence generation models of Rashkin et al. (2018) instead, despite the equivalence with their encoder-decoder architecture, STEPS draws its predictions from the full English vocabulary rather than predicting the unigrams of the type vocabulary directly.

On another line of research, increasing attention to event understanding has been gained by downstream applications based on events. A non-exhaustive list includes, *inter alia*, Tomai and Forbus (2010); Mostafazadeh et al. (2017); Chaturvedi, Peng, and Roth (2017) for narrative prediction, Berant et al. (2014) for machine comprehension, Zhukov et al. (2019); Fried et al. (2020) for video segmentation. By means of STEPS, we provide a simple general-purpose architecture for event understanding which, besides its high performances achieved in intent prediction, can also be easily integrated into the aforementioned applications and those requiring commonsense reasoning based on chains of activities, thus potentially advancing the state of the art.

[2]Chen et al. (2020) use the WordNet (Miller 1995) meaning inventory.

3 Methodology

In this Section, we first introduce the Multi-axis Event Process Typing task formally, and then describe our sequence-to-sequence reformulation and autoregressive approach to solving it.

Multi-axis Event Process Typing

Given a process $\hat{p}$, composed of a sequence of n events $E_{\hat{p}} = e^{\hat{p}}_1, \ldots, e^{\hat{p}}_n$, the Multi-axis Event Process Typing task aims at classifying $\hat{p}$ along two different axes: the type of action involved in the process ($a_{\hat{p}}$) and the type of object the process seeks to affect ($o_{\hat{p}}$). Following the original formulation by Chen et al. (2020), we treat both the action and object types as free-form labels. Finally, leveraging this formalism, we can describe any system performing the Multi-axis Event Process Typing task as a function f such that $f(E_{\hat{p}}) \rightarrow (a_{\hat{p}}, o_{\hat{p}})$.

Sequence-to-Sequence Formulation

While Chen et al. (2020) tackle the Multi-axis Event Process Typing as a learning-to-rank problem that assigns to $E_{\hat{p}}$ a sorted list of action and object types, we put forward a new formulation as a sequence generation problem. In this formulation, we require a model to *generate* a sequence of tokens from which we can extract the action and object types when it is provided with a sequence of events as input. More formally, the model is fed a sequence of events m defined as:

$$
\begin{aligned}
m = &<e1>, t^1_{e^{\hat{p}}_1}, \ldots, t^i_{e^{\hat{p}}_1}, \ldots t^k_{e^{\hat{p}}_1}, </e1>, \\
&\ldots, \\
&<eh>, t^1_{e^{\hat{p}}_h}, \ldots, t^j_{e^{\hat{p}}_h}, \ldots t^l_{e^{\hat{p}}_h}, </eh>, \\
&\ldots, \\
&<en>, t^1_{e^{\hat{p}}_n}, \ldots, t^g_{e^{\hat{p}}_n}, \ldots t^o_{e^{\hat{p}}_n}, </en>.
\end{aligned}
$$

where $t^j_{e^{\hat{p}}_h}$ represents the j-th token of the h^{th} event of the process $\hat{p}$ and the special tokens ($<eh>, </eh>$) are the delimiters for the tokens of the h-th event. Next, a model is required to produce any target sequence $\hat{s}$, from which one can extract the action and object types without supervision. For instance, a valid output sequence to predict the action type COOK and the object type PASTA can be: *"how to <a> cook <o> pasta </o>"*, since the sequence contains the action and the object types and both can be extracted by leveraging the special markers [*<a>*, **, *<o>*, *</o>*].

With this formulation, we can train any sequence-to-sequence model to learn the factorized probability:

$$
p(\hat{s}|m) = \prod_{j=2}^{|\hat{s}|} p(\hat{s}_j|\hat{s}_{1:j-1}, m) \tag{1}
$$

by minimizing the cross-entropy loss with respect to $\hat{s}$.

Posing Multi-axis Event Process Typing as a sequence-to-sequence task brings several advantages:

- The models do not depend on any external inventory for the action and object types, a counter-intuitive binding when considering free-form labels.
- There is no need to store possibly large dense representations for all the entities in the inventory as is necessary in Chen et al. (2020).
- The models benefit more from comparable pretrained language models – as we will show later in this paper – than when using a ranking objective.

4 Experimental Setup

In this Section we detail the experimental setting in which we train and evaluate our approach for Multi-axis Event Process Typing. In addition, we also describe the dataset we used, the sequence-to-sequence model we proposed alongside its training hyperparameters, and finally, our main comparison systems along with several STEPS variants.

Dataset

To support the Multi-axis Event Process Typing task, Chen et al. (2020) constructed an automatic dataset by scraping the *how-to* guides from wikiHow.[3] Each page in wikiHow corresponds to a complete guide divided into multiple steps on how to do something (e.g., the page "How to Cook Pasta" discusses the steps required to prepare pasta). Thus, to create a dataset instance, they used the guide steps as the sequence of events and leverage a Semantic Role Labeling (SRL) system to extract the principal action and object types from the title of the wikiHow article. Following this approach, the authors collected $62{,}277$ different event processes with $1{,}336$ action types and $10{,}441$ object types. Interestingly, most of the dataset is composed of action and object types as few-shot cases. Indeed, 68.3% of action types and 88.2% of object types occur fewer than 10 times across all the processes. Moreover, Chen et al. (2020) applied several heuristics to filter out both single events and entire processes to refine the whole dataset.[4] In order to assess the capabilities of our model to handle noisy data and extract meaningful information from them, we also train and evaluate STEPS on the raw wikiHow data where no processing steps have been applied. This results in a larger dataset of $108{,}027$ instances with longer processes, i.e., containing more steps.[5] Finally, the only modification that we apply to the dataset is in the output sequences that we wish to predict. Indeed, following our formulation, we have to produce a sequence containing both the action and the object types from which they can be extracted. To this end, our output sequence follows the title format of the wikiHow guides (e.g., the page "How to Cook Pasta"), with the addition of the special characters to highlight the action and object types (e.g., "How to <a> Cook <o> Pasta </o>"), using the extracted types by Chen et al. (2020). Finally, following Chen et al. (2020), we train all the models on a split containing 80% of the whole dataset, and use as validation and test sets two equally sized partitions which include the remainder of the data.

STEPS Model

STEPS builds on top of BART (Lewis et al. 2020), a sequence-to-sequence architecture pretrained with denoising objectives. In fact, BART has been trained on a large amount of English text, composed of corpora with varying sizes and domains, containing books, stories, news, Web content, and Wikipedia articles. This provides a wealth of information that could be helpful in our case due to the numerous few-shot types in the dataset. We analyze our training strategies when fine-tuning both BART Base (139M parameters) and Large (406M parameters).[6] We train STEPS with Adagrad (John, Elad, and Yoram 2011) for a maximum of $300{,}000$ steps with a learning rate of 2×10^{-5}, batches of 800 tokens and a gradient accumulation of 10 steps. Finally, we evaluate STEPS performances every $2{,}000$ updates, and we interrupt the training if no improvements are observed in the validation set for 3 consecutive evaluations. At prediction time we use beam decoding with a beam size of 5. The experiments are carried out using an Nvidia GeForce RTX 2080ti.

Comparison Systems

In order to assess the capabilities of our sequence-to-sequence formulation, we compare STEPS models against several systems. We consider in our comparison:

- S2L: the sequence-to-label generators (Rashkin et al. 2018) initialized with different encoders, introduced as baselines in Chen et al. (2020). S2L models are architecturally equivalent to encoder-decoder networks, but they are trained to directly predict unigrams of the types vocabulary. We compare with three different encoder initializations using: RoBERTa (S2L-RoBERTa), a BiGRU RNN (S2L-BiGRU), and a mean-pooling encoder (S2L-mean-pool).
- P2GT: the best-performing systems under the Process Typing with Gloss Knowledge (P2GT) framework proposed in Chen et al. (2020). We include in the evaluation both the systems trained on the glosses assigned via the Most Frequent Sense (MFS) and the glosses predicted by the external WSD model they use. Furthermore, we report the performances of their systems both when the action types and the object types are learned jointly and separately. We compare with a total of four different models: Single P2GT-MFS (Single training + MFS), Single P2GT-WSD (Single training + Dedicated WSD system), Joint P2GT-MFS (Joint training of actions and objects + MFS), and Joint P2GT-WSD (Joint training of actions and objects + Dedicated WSD system).

[3] https://www.wikihow.com

[4] We want to highlight that Chen et al. (2020) do not present trials with the raw dataset; therefore, it is unclear whether the preprocessing performed was essential.

[5] There are on average 2 steps in the preprocessed dataset and 5 steps in the raw one.

[6] We use the pretrained weights of both models made available by the Transformers library (Wolf et al. 2020)

	Model	Action Typing			Object Typing		
		MRR	recall@1	recall@10	MRR	recall@1	recall@10
S2L	S2L-mean-pool	3.72	1.96	5.95	1.01	0.80	1.66
	S2L-BiGRU	7.94	4.40	12.71	4.20	2.72	6.19
	S2L-RoBERTa	8.36	5.31	14.69	4.88	3.24	8.10
P2GT	Single P2GT-MFS	24.10	19.67	32.40	13.71	8.86	23.09
	Single P2GT-WSD	25.83	19.93	37.50	14.19	9.32	24.84
	Joint P2GT-MFS	28.57	20.63	43.14	15.26	10.62	25.01
	Joint P2GT-WSD	29.11	21.21	42.84	15.70	11.07	25.51
STEPS BB	Original glosses	40.54	31.27	59.87	18.73	13.42	29.15
	ESC glosses	41.03	31.92	64.02	20.57	15.23	30.09
	No glosses	39.73	30.68	60.56	20.19	15.11	30.11
	Raw data and ESC glosses	49.39	38.22	72.16	32.58	24.24	49.86
	Raw data without glosses	50.49	39.43	72.87	33.01	24.41	49.93
STEPS BL	Original glosses	40.76	31.54	61.05	19.89	14.29	31.23
	ESC glosses	42.09	33.04	61.91	19.74	14.18	31.33
	No glosses	41.22	31.97	62.27	21.11	15.51	31.42
	Raw data and ESC glosses	50.17	38.92	73.08	33.14	25.12	53.86
	Raw data without glosses	**51.55**	**40.38**	**74.34**	**35.84**	**26.28**	**54.14**

Table 1: Results in the test set in terms of MRR, recall@1, and recall@10. S2L denotes the sequence-to-label models of Rashkin et al. (2018), P2GT denotes the models of Chen et al. (2020), STEPS_{BB} and STEPS_{BL} denote our system based on BART Base and BART Large, respectively. We report the performance of S2L and P2GT as calculated by Chen et al. (2020). The best scores per row block are underlined while the best across the board are shown in **bold.**

- STEPS: the various models trained with our sequence-to-sequence formulation obtained by varying: i) the underlying Transformer architecture, namely BART Base (STEPS_{BB}) and BART Large (STEPS_{BL}); ii) the training data, i.e., the raw (Raw data) or the preprocessed data by Chen et al. (2020); and finally, iii) whether the models are required to produce the action/object definitions or *only* the types. In fact, the usage of glosses is crucial for Chen et al. (2020), therefore here we analyze whether STEPS might also benefit from this extra information.[7] To this end, we train STEPS to produce the glosses used by Chen et al. (2020) (Original glosses), those obtained by applying ESCHER (Barba, Pasini, and Navigli 2021), i.e., the state-of-the-art model in WSD at the moment of writing (ESC glosses), and no glosses at all (No glosses).

As regards the evaluation metrics, we follow Chen et al. (2020) and use three ranking measures: Mean reciprocal rank (MRR), Recall at 1 (recall@1), and Recall at 10 (recall@10). Even with our sequence-to-sequence formulation, we can easily provide a ranking of the action and the object types by producing the top-k most likely sequences for Equation 1.

[7]In order to enable the model to produce the definitions, we just append them in the output sequence along with their respective action or object type.

5 Results

In Table 1, we report the performance of all the aforelisted systems. First, all the sequence-to-label models (S2L row block) severely underperform in the task when compared to the other two formulations. This result is, potentially, due to the ultra-fine-grained classes that, combined with the dataset distribution, make it difficult to grasp the types by relying solely on the training data and without any external knowledge (e.g., glosses from an external sense inventory).

Secondly, we can see that all the models under the STEPS_{BB} row block, i.e., STEPS_{BB} trained under different scenarios, perform significantly better than even the best model under the P2GT row block. This is especially interesting because the Joint P2GT-WSD model and STEPS_{BB} are trained with the same data and glosses from Chen et al. (2020). Moreover, their underlying pretrained models have a comparable number of parameters (RoBERTa base 125 million and BART Base 139 million parameters). This result strongly supports the claim that our sequence-to-sequence formulation benefits more from the pretrained knowledge of underlying models and, therefore, is particularly suited for the semantic typing of event processes.

As mentioned earlier in Section 4, we analyze whether STEPS models benefit from the additional raw data prior to the preprocessing of Chen et al. (2020). To this end, we train the models on the dataset without discarding the noisy events and processes (Raw data). The results show

that our models are fully capable of extracting information from noisy data and taking advantage of it, outperforming their counterparts by a large margin, i.e., when trained on the preprocessed data. This is, in fact, a characteristic of the models that allows them to deal with and benefit from noisy and raw text while, at the same time, eliminating the need to rely on any filtering pipeline. Furthermore, we investigate whether extra definitional information in the target sequences is beneficial. To this end, we train STEPS to produce not only the action and the object types, but also their definitions, using those provided by Chen et al. (2020) and by applying ESCHER. Interestingly enough, when using the processed dataset, producing the definitions yields higher performances than when no glosses are included. When training on raw data instead, the performances of the models that leverage the glosses in their output sequences decrease compared to the models that predict *only* the types. This is possibly due to the fact that, when receiving less information in input, the glosses help in augmenting the output, thus influencing the predicted types; on the other hand, when the number of events in the input sequence is bigger and the training instances are augmented, the burden of producing longer sequences may negatively affect the search space of beam decoding, thereby reducing the benefits of including more information in the target sequences. Finally, we repeat all the experiments by replacing BART Base with BART Large as the underlying pretrained model. Unsurprisingly, STEPS_{BL} achieves consistenly higher performance than STEPS_{BB}, while showing the exact same behaviors in the different training regimes. Even if the performance gap is relatively narrow when comparing STEPS_{BB} and STEPS_{BL} under the standard setting, STEPS_{BL} brings advantages in more challenging training settings (see Section 6).

6 Analysis and Discussion

Here we perform a thorough evaluation of STEPS in different synthetic training and testing scenarios. Our analysis is mainly motivated by the few-shot nature of the dataset within which, we recall from Section 4, 68.3% of action types and 88.2% of object types occur fewer than 10 times across all the processes. Moreover, we further enrich our analysis by evaluating the models in settings where event sequences' natural order and completeness are compromised. In particular, we observe the behavior of the models when:

- downsizing the training set while maintaining the original distribution,
- subsampling the training set according to a maximum number of instances for each action or object type, thus changing the original distribution,
- partitioning the test set as per the actions' frequency,
- varying the order of events per process at inference, and
- varying the number of events per process at inference.

Train Downsizing

We train the best versions of STEPS_{BB} and STEPS_{BL} (see Table 1) on samples comprising 10%, 25%, 50% and 75% of the instances drawn randomly from the original training set and report their MRR scores on the test set. The results are visualized in Fig. 1. First of all, we observe two expected behaviours: i) more data bring higher performance for both STEPS_{BB} and STEPS_{BL}, with a steeper slope on average up to the first 50% of the dataset and a smoother one for the remaining 50%, and ii) STEPS_{BL} consistently achieves higher performance than STEPS_{BB}, with a larger gap in the smaller training splits. Interestingly, action typing is more affected by the train downsizing than object typing is. This may result from the fact that the distribution of action types in the dataset is very skewed towards the most frequent ones, and hence, downsizing the training dataset leads to sampling an even lower number of the rare types.

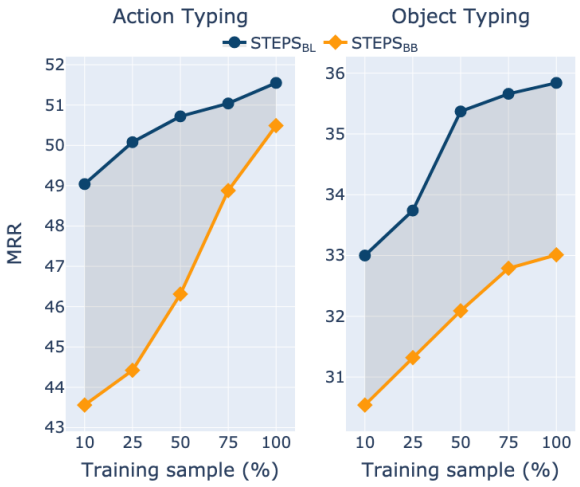

Figure 1: Comparison of STEPS_{BB} and STEPS_{BL} on different training splits: Action typing is more affected by train downsizing than object typing is.

Few-shot Learning

In this Section we perform several experiments mimicking few-shot training conditions, as well as evaluating STEPS variants on the test set partitions comprising challenging examples. First of all, we evaluate STEPS when trained on a maximum number of instances (k) for each action or object, separately. The results when varying k in a range 1-10 are shown in Fig. 2. Firstly, it is worth considering that sampling based on action frequency yields significantly smaller subsamples due to the remarkably diverse vocabulary sizes for action and object types.[8] Indeed, this is one of the main reasons behind the low results of the models when applying few-shot sampling on the action types, which: i) for each k, yields a MRR score that is, on average, 22 points lower in action typing and 5 points lower in object typing than when training on the whole training set, and ii) affects both action and object typing performances more than the few-shot setting according to object types. In fact, the few-shot on ob-

[8]The size of the training set sampled according to the action types frequency sampling for each k is only 13-18% of the training set sampled based on the object frequency for the same k.

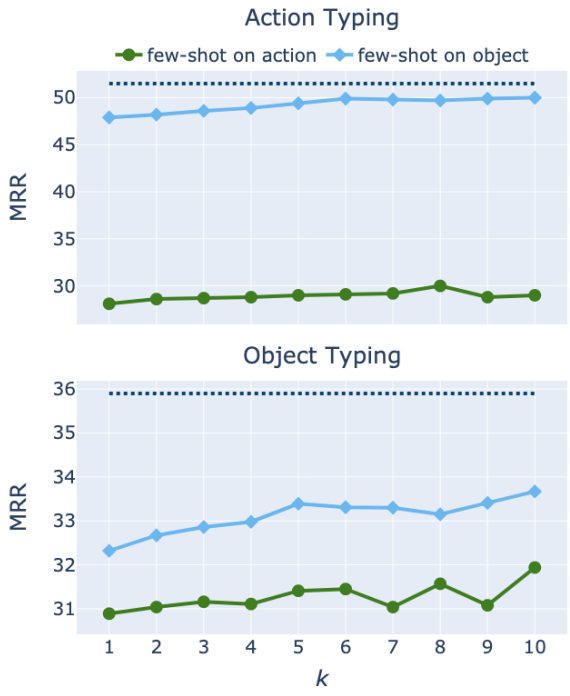

Figure 2: STEPS$_{BL}$ performance on action (upper) and object (lower) typing when trained on the few-shot setting on actions and objects separately: Changing the distribution of the training data significantly harms the performance.

ject types training affects the action and object typing performance by no more than 2 and 3 MRR points, respectively. The remarkable difference between the action typing performance in both settings, despite the training set size disparity, suggests that even when decreasing per-object examples in the dataset, the frequency of action types remains large enough to avoid a large drop in performances. On the other hand, it may seem surprising that, when considering the action predictions and varying the number of instances per action type, we have almost a constant score $\forall k \in [1, 10]$. Probably, this is due to the more skewed distribution towards the most frequent types when considering actions. Indeed, the top 100 frequent actions constitute roughly 80% of the total instances in the dataset. This makes the subsampled datasets particularly out of distribution with respect to the test set, thus explaining the low performances regardless of the increase in the dataset size when varying k.

In addition to training set sampling, we also construct synthetic test sets on which we evaluate STEPS. Following Chen et al. (2020), we split the test set according to action type frequency and compare results by P2GT with STEPS$_{BB}$ and STEPS$_{BL}$ for the top 100 frequent types, one-shot types and others. These results are shown in Fig. 3. Firstly, both STEPS$_{BB}$ and STEPS$_{BL}$ achieve better performance than P2GT in all the splits. While our system easily surpasses the P2GT baseline on the most frequent types, the superior performance in the less frequent types suggests

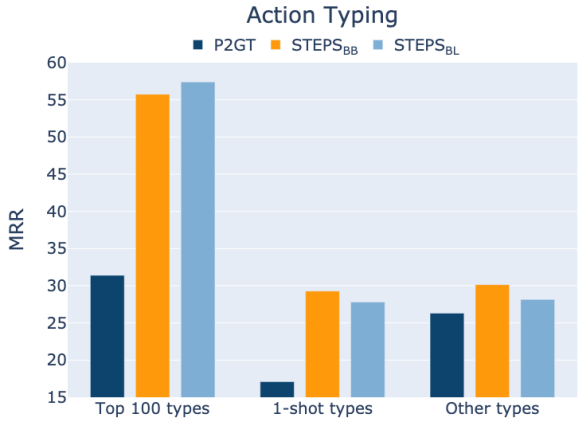

Figure 3: Comparison of STEPS models and P2GT on action typing on different test splits: Both STEPS models outperform P2GT, yet have the similar behaviour across splits.

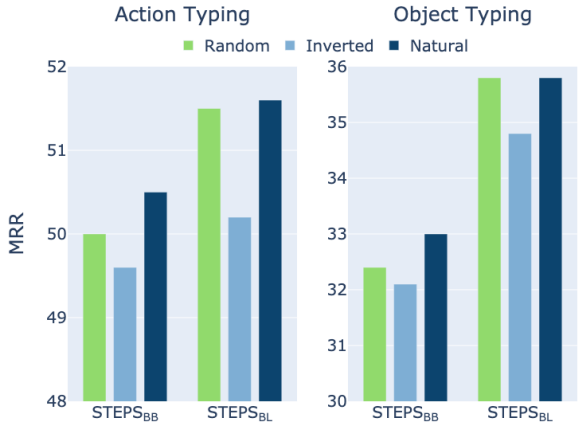

Figure 4: STEPS models behavior when changing the order of the events: Event reordering does not significantly affect the action and object typing performances.

that our sequence-to-sequence formulation allows a system to benefit more from pretrained language models – of comparable sizes – leading to a better generalization. Interestingly, except for the top-100 split, STEPS$_{BB}$ outperforms STEPS$_{BL}$ in the one-shot and other settings. This indicates that larger models may be liable to overfit more on the most common labels during training.

Compromising Event Processes

Here we analyze the effect of compromising the events at inference time by i) changing events' order, and ii) removing the events from the process sequence. In Fig. 4 we report the action and object typing performances of STEPS$_{BB}$ and STEPS$_{BL}$ on the test set when we shuffle or invert the event steps, along with the performances on the non-compromised

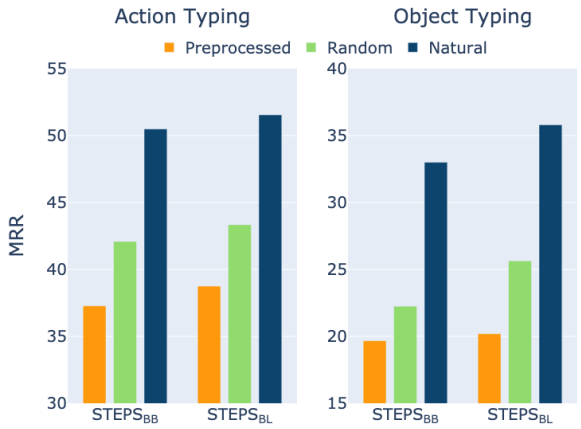

Figure 5: STEPS models behavior when removing events from the sequence: Event removal significantly affects action and typing performances.

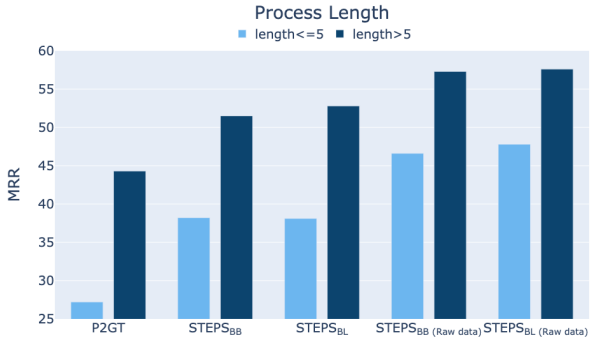

Figure 6: Comparison of STEPS models and P2GT on test splits according to process length: All systems perform better on longer process and STEPS$_{BL}$ trained on Raw data achieves the highest performances.

test set (Natural). Surprisingly, the performance gap between the natural ordered processes and the compromised ones is not significant, even though inverting the steps appears to be more harmful. This result is probably because the task is about inferring the overall goal instead of predicting what happens next, in which case order might be more influential. For instance, considering our running example composed of the following steps, 1. *boil the water* 2. *add salt* 3. *add pasta* ... 6. *add sauce as per preference*, a human might infer that the intent is that of COOKING PASTA even if one *adds the pasta* before *adding the salt*. We believe that further research in this direction might be interesting for future works.

Furthermore, we analyze how completely removing some of the steps from the event sequence affects the performance. This evaluation is particularly interesting as it is similar to the phenomenon that occurs during the preprocessing of the dataset by Chen et al. (2020) which, we recall from Section 4, decreases the average length of processes from 5 (raw dataset) to 2 events. For this test we use the test split from the raw dataset (Natural), the preprocessed dataset of Chen et al. (2020) (Preprocessed) and a synthetically constructed test set in which we filter out steps following a similar distribution to those of the preprocessed dataset.[9] In Fig. 5 we show the performances of STEPS$_{BB}$ and STEPS$_{BL}$ in these settings. As we can see, filtering out steps drastically undermines the performance. Interestingly, removing steps randomly attains a higher performance than that achieved in the preprocessed dataset by Chen et al. (2020). This suggests that the heuristics used by the latter are harmful, as they not only remove more than half of the steps in an event sequence, but also it is likely that the removed steps are essential for the event understanding task. Therefore, removing steps from processes degrades the performance as: i) it provides less information in the input and thus less context to learn from, and ii) it breaks the procedural flow of processes.

Finally, continuing on from the previous analysis, we observe the effect of processes length on the performance. Following Chen et al. (2020), we compare the action typing performance of STEPS and P2GT in the splits of the test set comprised of the processes that contain at least or higher than 5 events. The results are shown in Fig. 6. As expected, the performance on longer processes is higher than that on shorter processes for all the models, with STEPS$_{BL}$ model trained on Raw data performing best across the board.

7 Conclusion

In this work we presented a novel formulation for the Multi-axis Event Process Typing task as a sequence generation problem and put forward STEPS, a sequence-to-sequence approach that achieves state-of-the-art performances under different settings and scenarios. Not only does STEPS outperform its competitors on the evaluation suite made available by Chen et al. (2020), but we also demonstrated that it successfully handles the *raw* version of the dataset, being able to extract the core information and improving the performances by more than 10 points in terms of MRR for both action and object typing. Furthermore, the experiments highlight that the models' performances are affected more by limiting the number of instances per type than by downsizing the training set following the original distribution. Moreover, we found that the ordering of the events in a process does not represent an indispensable item of information for the models, which are fully capable of correctly predicting both the object and the action types even when the sequence of events is shuffled. To the contrary, the models are sensitive to missing information when tested on a randomly pruned version of the dataset, with a drop of more than 13 MRR points on action and object typing. As future work, we would like to first improve the applicability of the task by extracting object and action types from raw text instead of from ordered event sequences. Then, we would like to analyze the boost in performances from the perspective of the downstream tasks, rather than from just the ranking scores. We release STEPS at https://github.com/SapienzaNLP/steps.

[9] We sample the fraction of events to drop from a Gaussian distribution with mean 0.6 and variance 1.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487 and the ELEXIS project No. 731015 under the European Union's Horizon 2020 research and innovation programme.
This work was partially supported by the MIUR under the grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science of Sapienza University.

References

Baker, C. F.; Fillmore, C. J.; and Lowe, J. B. 1998. The Berkeley FrameNet Project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, 86–90. Montreal, Quebec, Canada: Association for Computational Linguistics.

Barba, E.; Pasini, T.; and Navigli, R. 2021. ESC: Redesigning WSD with Extractive Sense Comprehension. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4661–4672. Online: Association for Computational Linguistics.

Berant, J.; Srikumar, V.; Chen, P.-C.; Vander Linden, A.; Harding, B.; Huang, B.; Clark, P.; and Manning, C. D. 2014. Modeling Biological Processes for Reading Comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1499–1510. Doha, Qatar: Association for Computational Linguistics.

Chambers, N.; and Jurafsky, D. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of ACL-08: HLT*, 789–797. Columbus, Ohio: Association for Computational Linguistics.

Chaturvedi, S.; Peng, H.; and Roth, D. 2017. Story Comprehension for Predicting What Happens Next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1603–1614. Copenhagen, Denmark: Association for Computational Linguistics.

Chen, M.; Zhang, H.; Wang, H.; and Roth, D. 2020. What Are You Trying to Do? Semantic Typing of Event Processes. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, 531–542. Online: Association for Computational Linguistics.

Das, D.; Chen, D.; Martins, A. F. T.; Schneider, N.; and Smith, N. A. 2014. Frame-Semantic Parsing. *Computational Linguistics*, 40(1): 9–56.

Fried, D.; Alayrac, J.-B.; Blunsom, P.; Dyer, C.; Clark, S.; and Nematzadeh, A. 2020. Learning to Segment Actions from Observation and Narration. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2569–2588. Online: Association for Computational Linguistics.

John, D.; Elad, H.; and Yoram, S. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(61): 2121–2159.

Kipper Schuler, K.; Korhonen, A.; and Brown, S. 2009. VerbNet overview, extensions, mappings and applications. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Tutorial Abstracts*, 13–14. Boulder, Colorado: Association for Computational Linguistics.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880. Online: Association for Computational Linguistics.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11): 39–41.

Mostafazadeh, N.; Roth, M.; Louis, A.; Chambers, N.; and Allen, J. 2017. LSDSem 2017 Shared Task: The Story Cloze Test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, 46–51. Valencia, Spain: Association for Computational Linguistics.

Radinsky, K.; and Horvitz, E. 2013. Mining the Web to Predict Future Events. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, 255–264. New York, NY, USA: Association for Computing Machinery. ISBN 9781450318693.

Rashkin, H.; Sap, M.; Allaway, E.; Smith, N. A.; and Choi, Y. 2018. Event2Mind: Commonsense Inference on Events, Intents, and Reactions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 463–473. Melbourne, Australia: Association for Computational Linguistics.

Tomai, E.; and Forbus, K. 2010. Using Narrative Functions as a Heuristic for Relevance in Story Understanding. In *Proceedings of the Intelligent Narrative Technologies III Workshop*, INT3 '10. New York, NY, USA: Association for Computing Machinery. ISBN 9781450300223.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Le Scao, T.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Online: Association for Computational Linguistics.

Zacks, J. M.; Speer, N. K.; Swallow, K. M.; Braver, T. S.; and Reynolds, J. R. 2007. Event perception: a mind-brain perspective. *Psychological bulletin*, 133 2: 273–93.

Zhang, H.; Chen, M.; Wang, H.; Song, Y.; and Roth, D. 2020. Analogous Process Structure Induction for Sub-event Sequence Prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Process-*

ing (EMNLP), 1541–1550. Online: Association for Computational Linguistics.

Zhukov, D.; Alayrac, J.-B.; Cinbis, R. G.; Fouhey, D.; Laptev, I.; and Sivic, J. 2019. Cross-Task Weakly Supervised Learning From Instructional Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.