Extracting relations concerning personal-attributes from dialogue

Zhilin Wang
University of Washington
zhilinw@uw.edu

Abstract

We propose a novel relation extraction task that aims to extract personal attributes (such as what people like and have) from dialogue. These personal attributes can subsequently be useful in dialogue modelling or personalizing recommendations. We treat this task as a language modelling task, achieving comparable performance to Wadden et al. (2019), which achieved SOTA performance in other relation extraction tasks. At the same time, our model can also extract relations in which head and tail entities are not contained in the original sentence due to synonyms/paraphrases or commonsense inferences, which are common in everyday conversation.

1 Introduction

Personal-attributes are information about a person such as what they like, what they own, and what their favorite things are. These attributes are commonly revealed during social dialog, allowing people to know more about one another. However, limited research has been done on how dialog agents can effectively extract and represent such personal attributes beyond natural language sentences (Zhang et al., 2018). Knowledge graph triples are a promising alternative approach to represent such information, given their success in representing factual relations. They are a scalable approach to represent potentially thousands (or more) of personal attributes, while maintaining ease of access to the information and the ability to reason with such attributes (Hogan et al., 2021).

Not only can these personal attribute triples improve dialogue modelling (Majumder et al., 2020; Moon et al., 2019), they can also support dialogue agents to personalize conversations with a user and make more suitable recommendations (Balog and Kenter, 2019; Li et al., 2014). For instance, the triple `(i, dislike, meat)` can improve restaurant suggestions. While such knowledge graphs triples can offer significant advantages in personal-attributes representation, a key bottleneck lies in extracting these triples from natural language. To address this bottleneck, we propose a supervised learning task to extract personal-attributes knowledge graphs triples from natural language, by re-purposing the DialogNLI dataset (Welleck et al., 2019). We also introduce a model for the task, based on a causal language model.

2 Task definition

Research question: How can personal-attributes be extracted from natural language using a language-modelling training objective?

Given a sentence S, consisting of tokens t_1 to t_n, we want to obtain a personal-attribute relation triple in the form of (head_entity, relation, tail_entity), with examples in Table 1. The head_entity and tail_entity each comprises of 1 or more tokens (called t_1^{head} to t_m^{head} and t_1^{tail} to t_k^{tail} respectively), which may not overlap with t_1 to t_n. This is because some of the head and tail entities are either synonyms of phrases used in the original sentence or derived based on commonsense reasoning used by annotators (Sap et al., 2019). This is in contrast to common relation extraction tasks, in which both the head and tail entities can be found in t_1 to t_n (Wadden et al., 2019), making our task formulation significantly more challenging. Relation consists of only one token (called t^{reln}), which is one of 60 special tokens, that does not overlap with t_1 to t_n.

3 Approach

During training, we processed our input by concatenating the sentence, head_entity tokens, relation token as well as tail_entity tokens alongside special control tokens ([HEAD], [RELN], [TAIL]) to demarcate the start of each element of the triple as follows:

new_seq[t_1, ..., t_j] = t_1, ..., t_n, [HEAD], t_1^{head}, ..., t_m^{head}, [RELN], t^{reln}, [TAIL], t_1^{tail}, ..., t_k^{tail}

For instance: oh , i like to read to my 2 cats . [HEAD] i [RELN] [like_animal] [TAIL] cats

We adopt the causal language modelling objective of max $\prod_{i=1}^{j} P(t_i|t_1, ..., t_{i-1}; \theta)$ over t_1 to t_j.

During evaluation, we generate a sequence with t_1 to t_n as context and use the tokens in between the special control tokens as the values for each field within the triple.

4 Related work

4.1 Personal-attributes relation extraction

Most work on extracting personal attributes from natural language (Tigunova et al., 2019, 2020; Pappu and Rudnicky, 2014) employed weak supervision approaches using heuristics and hand-crafted templates. In contrast, we use a strong supervision approach in which triples were manually annotated. Yu et al. (2020); Li et al. (2014) also attempted to extract relations from dialogue using a strongly supervised paradigm. However, they focused on demographic attributes as well as interpersonal relationships, which contrasts with our focus on what people own and like.

4.2 Causal language models for relation extraction and dialogue state tracking

Alt et al. (2019) used the original GPT model for relation extraction. However, instead of using only the language model loss, they included an additional classification loss for their relation token. This is feasible for their task because they are only interested in identifying a relation out of around 30 possible options. On the other hand, our task seeks to identify not only the relation but also the head and tail entities, which have significantly large search spaces (in the thousands), making classification impractical.

Hosseini-Asl et al. (2020) trained a GPT-2 model for dialogue state tracking. This task seeks to extract information to perform slot-filling (in slots such as taxi destination or hotel location). Our task is similar in that it also seeks to extract structured information but differs in that dialogue state tracking only involves extracting information that are found in the original utterance.

5 Methods

5.1 Dataset

The dataset is re-purposed from the DialogNLI dataset (Welleck et al., 2019). The dataset contains sentences relating to personal attributes as well as a manually annotated triple paired with each sentence. Examples are shown in Table 1. The three largest groups of relations are: a. Has_X (where X = hobby, vehicle, pet) b. Favourite_Y (where Y = activity, color, music) c. Like_Z (where Z = read, drink, movie) and most common relations can be found in Figure 7 in the appendix. We follow train/dev splits in the original dataset with descriptive statistics in Table 2.

To make our results comparable to a baseline model (Wadden et al., 2019) that can only identify head and tail entities if they were found in the initial sentence, we split our data into two subsets. `within-sentence` contains sentences/associated triples in which both the head and tail entity are contained in the sentence as contiguous spans, while the complementary set is `not-within-sentence`. We conduct experiments with 3 sets of data: the whole dataset (called `all`), the within-sentence subset and the not-within-sentence subset.

5.2 Dataset analysis

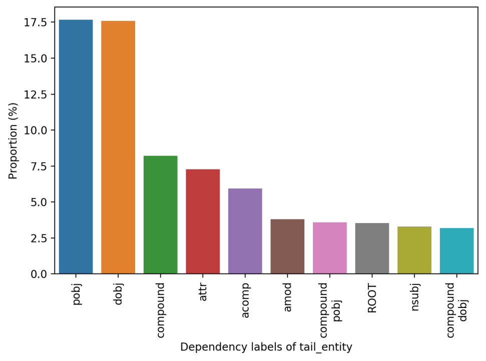

Figure 1: Bar plot for 10 most common dependency labels of tail_entities

To better understand the dataset, we analyze the syntactic roles that head and tail entities play in the sentences. Because the diversity of head_entities is much more restricted (93.3% are 'I'), this analysis will focus on the tail_entities. Most of this analysis will focus on the within-sentence data subset since the head and tail entities can be found in the original sentences. However, we will also utilise some

Sentence	Triple	Data subset
i work as a receptionist at a lawyers office.	(i, [has_profession], receptionist)	within-sentence
oh , i like to read to my 2 cats .	(i, [like_animal], cats)	within-sentence
i play guitar in my spare time .	(i, [has_ability], play instrument)	not-within-sentence
my favorite band is imagine dragons .	(i, [favorite_music_artist], imagine dragons)	not-within-sentence
my ultimate goal would be calling a ball game .	(i, [want_job], commentator)	not-within-sentence

Table 1: Examples of sentences and paired triples

	Train	Dev
no. of samples	58790 (45% within-sentence)	6263 (47% within-sentence)
no. of words in sentence	12.50 ($\sigma = 4.24$)	12.64 ($\sigma = 4.00$)
no. of unique head_entity	155 (93.3% 'i')	30 (93.3% 'i')
no. of unique relations	60	55 (100% in train)
no. of unique tail_entity	4832	837

Table 2: Descriptive statistics

examples from the not-within-sentence data subset to compare how words in the original sentences can be useful to determine the tail_entities. We use dependency parses of sentences to understand the relationship between words within tail_entities and the sentence ROOT. Dependency parsing was chosen because it is a well-studied syntactic task (Nivre et al., 2016) and has been shown to contribute to the relation extraction task (Zhang et al., 2017). Dependency parses and labels associated with each dependent word[1] were identified using a pre-trained transformer model by spaCy[2].

From Figure 1, we can tell that a significant proportion (around 17.5%) of tail_entities are either objects of prepositions (pobj) or direct objects (dobj). This is followed by compound words (compound), attributes (attr) and adjectival complements (acomp) with around 7% each. This suggests that there is a diversity of grammatical roles that tail_entities play in the original sentence and therefore this task is likely difficult to tackle using hand-crafted patterns.

To illustrate this, Figure 2 shows an example of a tail_entity (Alabama) that plays the role of an object of preposition. Comparing this to Figure 3, a dependency parse of a sentence that does not contain the tail_entity (university as in (I, [attend_school], university)), the word that suggests university to be the tail_entity (i.e. school) also plays a similar syntactic role as the object of the predicate. This suggests that even for sentences that do not contain the tail_entity, it is likely that they can be inferred (by synonym/hypernym/hyponym/commonsense inferences) from certain words in the original sentence in a similar way as tail_entities that are contained in the original sentence. This means that the first step of resolving tail_entities that are not contained in the original sentence can be conceptualized as seeking for pseudo-tail_entities that are contained in the original sentence, before inferring tail_entities from the pseudo-tail_entities. However, given the range of possible inferences that can be made through synonyms/hypernyms/hyponyms/commonsense inferences, the second step might be extremely challenging.

5.3 Metrics

Precision/Recall/F1 were calculated following Wadden et al. (2019) in which a sample is only considered correct when all three elements (head_entity, relation and tail_entity) are resolved correctly. This makes our metrics significantly more stringent than those in Yu et al. (2020) and Alt et al. (2019), which consider samples correct when they resolve only the relation correctly (and are supplied with the ground truth head and tail entities). In addition, metrics were calculated for each element of the triple to provide insights into analysis. Validation was done every 0.25 epochs during training.

[1]Based on the schema from https://github.com/clir/clearnlp-guidelines/blob/master/md/specifications/dependency_labels.md

[2]https://spacy.io/

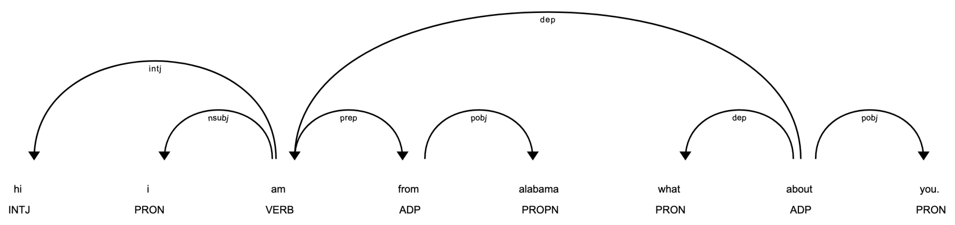

Figure 2: Example of dependency parse which contains the tail_entity

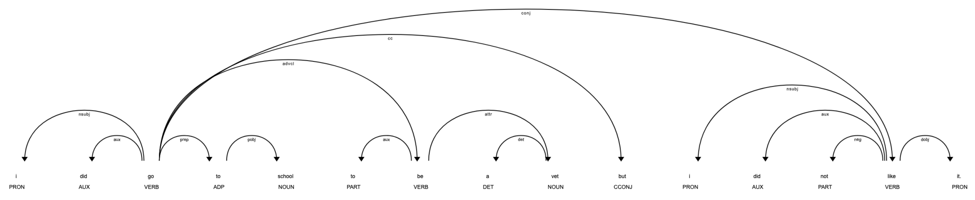

Figure 3: Example of dependency parse of sentence which does not contain the tail_entity

5.4 Baseline model

Wadden et al. (2019), a BERT based model with graph-propagation of related-entities, was used as our baseline model because it achieves SOTA performance in multiple Relation Extraction tasks. We adopted the hyperparameters[3] from the ACE05 task of extracting relations from news articles.

5.5 Model

5.5.1 Base model

GPT-2-small was used as a base model (details in Appendix 8.2). Additional special tokens including the special control tokens ([HEAD], [RELN], [TAIL]) as well as relation tokens were added into the tokenizer. Greedy decoding was used at inference time. We also made improvements on the base model, as described in the next section. Only improvements with significant positive impacts on performance are reported while less successful improvements can be found in the appendices 8.3.1, 8.3.2 and 8.3.3.

5.5.2 Improvement 1: Altering training objective

In a causal language model, tokens are generated one at a time during inference. Therefore, generating all three elements of a triple sequentially might cause early errors that mislead subsequent inferences. Therefore, we tried to generate each element of the triple separately using three models that do not share parameters (called `all_separate`).

[3]accessed from https://github.com/dwadden/dygiepp

5.5.3 Improvement 2: Optimizing generation hyperparameters

To find out if methods used to improve natural language generation could also improve performance on relation extraction, we attempted to use beam search with a beam size of $\{1, 5, 10\}$.

6 Results and discussion

6.1 Effects of improvements 1 and 2

As demonstrated in Table 3, generating each element independently improves performance relative to the base model. Much performance improvements can be attributed to gains on the tail_entity, since tail entities are more likely to be misled by mistakes made earlier when generating the head_entity and relation in the base model. Such differences might suggest that the best performance might be achieved by taking head_entity and relation predictions from the base model and the tail_entity predictions from the all_separate model. Beam search also has a significant positive effect on performance since beam search can alleviate the effect of making early mistakes when the correct token is not the most likely token but among the most likely tokens (Freitag and Al-Onaizan, 2017).

6.2 Performance relative to baseline model

Based on section 6.1, our best model is a composite model in which the head_entity and relation are derived from the base model while the tail_entity is derived from the all_separate model. For both models, beam search with num_beams = 10 was

	all			head_entity			relation			tail_entity		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Base model	38.2	15.5	18.8	95.4	90.7	92.5	44.3	37.1	38.4	55.2	25.1	29.8
Improvement 1: all_separate	**40.4**	16.3	**20.1**	**95.7**	89.2	91.7	38.7	37.9	34.2	**61.2**	**34.7**	**40.3**
Improvement 2: num_beams=10	36.1	**16.8**	20.0	95.1	**92.8**	**93.5**	**45.1**	**39.1**	**39.5**	54.2	27.6	31.8

Table 3: Performance of models with various improvements for all samples

	all			within-sentence			not-within-sentence		
	P	R	F1	P	R	F1	P	R	F1
Best model	**41.4**	**18.6**	**22.5**	**52.8**	30.8	36.3	**33.5**	**18.7**	**21.5**
Wadden et al. (2019)	-	-	-	41.5	**35.7**	**38.4**	-	-	-

Table 4: Overall performance (details of each element of the triple is available in Table 5)

used. Our best model is competitive to Wadden et al. (2019) with a significantly higher precision but significantly lower recall, and as a result, a slightly lower F1. The lower recall might be attributable to the fact that our model is generative compared to Wadden et al. (2019)'s discriminative model, meaning that our search space is significantly larger. Nonetheless, the generative nature of our model means that head and tail entities need not be contained in the original sentence, which makes our model less stringent. Instead, our model can handle synonyms/paraphrasing and commonsense inferences (Sap et al., 2019), which are common in conversations. For examples in Table 1, the vocation of "commentator" can be inferred from "calling a ball game" while the ability of "playing instrument" can be inferred as a hypernym of "playing guitar".

6.3 Error Analysis

This analysis is conducted with predictions made by the Best model on the `all` data subset. Based on Table 5, most mistakes are made either in the relation field (F1 = 39.5%) or the tail (F1 = 40.4%) compared to the head (F1 = 93.5%). Therefore, this analysis will focus on errors made in the relation field as well as the tail_entity. For each analysis, only the 10 most common values (both ground_truth and predicted) are considered.

6.3.1 Relation

Major sources of error come from relation tokens that have close semantic meanings. They are either synonyms of one another (e.g. [has_hobby] vs [like_activity]) or hyponyms/hypernyms of one another (e.g. [like_general] vs [like_activity]), as illustrated in Figure 4. Together, they account for 10.4% of errors. Such errors likely arise due to the way that the DialogNLI dataset (Welleck et al., 2019) was annotated. Specifically, annotators were asked to label a single possible triple given a sentence instead of all possible triples. Because of this, our evaluation metrics are likely to over-penalize our models when they generated reasonable triples that did not match ground truth. To overcome this challenge, the task can be re-formulated such that relations that are close in meaning can be mapped together to award credit to the model when the predicted triple is reasonable even if not completely matching the ground truth triple.

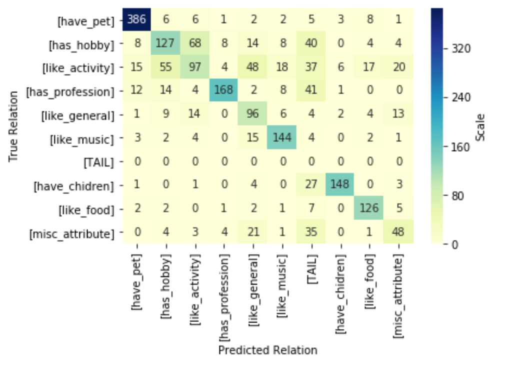

Figure 4: Confusion Matrix for 10 most common relations

Furthermore, there are also some generation artifacts such as the token [TAIL], which is the special control token to signify the start of the tail_entity. These account for 9.8% of errors. To overcome this issue, we can return multiple predicted relations and return the most likely relation that also does not contain special control tokens.

data	model	head_entity			relation			tail_entity		
subset		P	R	F1	P	R	F1	P	R	F1
all	Base model	95.4	90.7	92.5	44.3	37.1	38.4	55.2	25.1	29.8
	Best model	95.1	**92.8**	93.5	45.1	39.1	39.5	58.9	35.8	40.4
within-	Base model	**97.9**	87.2	92.2	**50.8**	44.3	45.9	68.3	39.4	46.7
sentence	Best model	97.8	92.7	**95.1**	49.6	**46.1**	**46.6**	**69.5**	**55.0**	**59.2**
not-within-	Base model	92.8	83.0	86.8	44.2	34.6	36.3	32.2	17.1	19.6
sentence	Best model	91.0	85.8	88.2	44.7	36.5	37.7	48.6	32.2	35.5

Table 5: Performance in each element of triple for all samples

6.3.2 Tail_entity

Part-of-Speech (POS) tags are useful for analysis because they encode the general category that a word belongs to, with both semantic and syntactic constraints on how the word can be used in a sentence. Universal POS tags[4] were identified using a pre-trained transformer model by spaCy[5]. We analysed POS tags for predicted tail_entities that do not exactly match the ground-truth tail_entities in terms of their tokens. As shown in Figure 6, only 27.9% were predicted tail_entities with the same POS tag(s) as the ground-truth tail_entities. Ground-truth tail_entities that are nouns are often predicted as entities that are adjectives, verbs or interjections (INTJ), as shown in Figure 5. Such mistakes also happen to a lesser extent for ground-truth tail_entities that are adjectives or verbs. In particular, interjections (words used as part of exclamations/emotional responses) are rare in ground-truth tail_entities (only 6 times) but are often predicted (257 times). A manual inspection of the data reveals that these interjections are often word fragments produced by sub-word tokenization used by GPT2.

Such observations suggest that the fine-tuned GPT2 might not always take into account the word category (i.e. POS tag) of tail_entity that it should predict. To overcome this issue, one possibility would be to re-rank possible tail_entities based on their POS tags. Alternatively, we can restrict the search space of tail_entities to only tokens that match the expected POS tag sequence. Taking this idea a step further would to be restrict search space to only entities with an expected entity_type. For instance, given the [has_profession] relation, the tail_entity is most likely a profession such as

[4]Based on the schema from http://universaldependencies.org/docs/en/pos/all.html

[5]https://spacy.io/

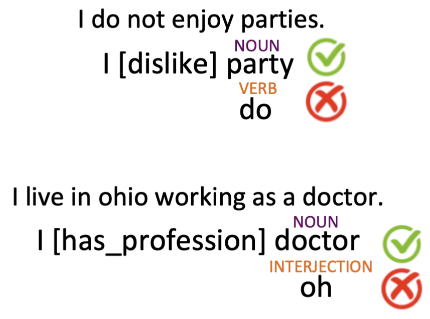

Figure 5: Examples for mistakes in predicting tail entities with POS tags

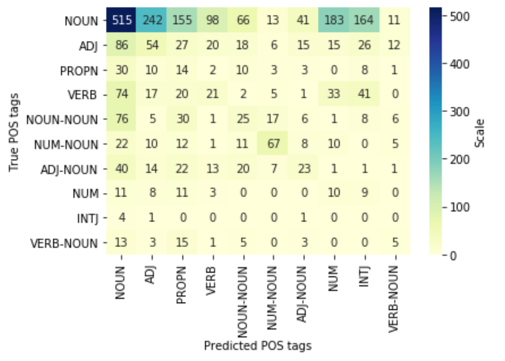

Figure 6: Confusion matrix for 10 most common POS tags of tail_entities

engineer/teacher. Such information flow between relations and tail_entities might also be modeled implicitly using graph neural networks(Wadden et al., 2019; Chen et al., 2020), without explicitly extracting POS tags and/or entity_types.

6.3.3 Overall analysis

Based on the analysis above, predictions of relations and tail_entities by the language model should be post-processed, instead of being directly used. Such post-processing include mapping relations with similar meanings and re-ranking based on similarity to expected POS tag sequences. Such post-processing can also be directly incorporated into the model itself, by reducing the original search space to only tail_entities that have an expected POS tag sequence (based on the relation) and/or entity_type. Graph neural networks (Wadden et al., 2019; Chen et al., 2020) might be a promising avenue to implicitly model such information flow between the relations and the the tail_entities, without explicitly extracting POS tags and/or entity_types. This way, a model can be trained without requiring such information to be defined in the dataset.

7 Conclusion

In conclusion, we propose a novel relation extraction task of extracting personal attributes from dialogue. We also present a competitive model for this task by treating this task as language modelling. Our model achieves comparable F1 to Wadden et al. (2019), a model that achieved SOTA performance on multiple relation extraction tasks. However, our model is useful for a wider range of labelled data because it does not require the head and tail entities to be contained within the original sentence. This allows our model to accommodate synonyms/paraphrases of as well as commonsense inferences from explicit statements, which are common in everyday conversation. Beyond using a language model directly for this task, integrating graph neural networks (Wadden et al., 2019; Chen et al., 2020) with language models can be promising to overcome some of the challenges that our analysis has revealed.

References

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Improving relation extraction by pre-trained language representations. In *Automated Knowledge Base Construction (AKBC)*.

Krisztian Balog and Tom Kenter. 2019. Personal knowledge graphs: A research agenda. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, page 217–220, New York, NY, USA. Association for Computing Machinery.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7521–7528.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *Proceedings of the First Workshop on Neural Machine Translation*.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge graphs.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue.

X. Li, G. Tur, D. Hakkani-Tür, and Q. Li. 2014. Personal knowledge graph population from user utterances in conversational understanding. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 224–229.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9194–9206, Online. Association for Computational Linguistics.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854, Florence, Italy. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Aasish Pappu and Alexander Rudnicky. 2014. Knowledge acquisition strategies for goal-oriented dialog systems. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and*

Dialogue (SIGDIAL), pages 194–198, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3027–3035.

Anna Tigunova, Andrew Yates, Paramita Mirza, and Gerhard Weikum. 2019. Listening between the lines: Learning personal attributes from conversations.

Anna Tigunova, Andrew Yates, Paramita Mirza, and Gerhard Weikum. 2020. CHARM: Inferring personal attributes from conversations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5391–5404, Online. Association for Computational Linguistics.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 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)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019. Dialogue natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741, Florence, Italy. Association for Computational Linguistics.

Dian Yu, Kai Sun, Claire Cardie, and Dong Yu. 2020. Dialogue-based relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4927–4940, Online. Association for Computational Linguistics.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45.

8 Appendix

8.1 Most common relations

8.2 Base model details

GPT2-small was accessed from HuggingFace Transformers library with 125M parameters, n_positions=1024, n_ctx=1024, n_embd=768, n_layer=12, n_head=12, dropout = 0.1. AdamW optimizer was used with $\alpha = 10^{-4}$, following a search within $\{10^{-5}, 5 * 10^{-5}, 10^{-4}\}$. Learning rate was linearly decayed (over a max epoch of 8) with 100 warm-up steps, following a search within {100, 1000}. Each training epoch took around 1 hour on an Nvidia V100 GPU with a batch size of 16.

8.3 Unsuccessful experiments

8.3.1 Improvement 1a: Altering training objective

The order in which the tokens are trained to be generated can also influence the effectiveness of the relation extraction. Our original approach generates the head entity, followed by the relation, and finally the tail entity. However, it is not known if generating other elements such as the relation or the tail entity first can improve relation extraction by altering information flow. It is plausible that generating the relation token first can reduce the search space for head/tail entities, given how certain relations can only be used with certain head/tail entities (e.g. [has_profession] and doctor and teacher but not game or strawberry). Therefore, we try all possible permutations of their order.

As demonstrated in Table 6, changing the orders of the sequence in which the head_entity, relation and tail_entity are generated did not significantly improve metrics relative to the base model. Noting that reln_head_tail performs closest to the base model (with better performance than the base model on the tail_entity) while tail_head_reln performs the worst, generating elements with fewer options (i.e. head_entity or relation) first is better than generating elements with more options (i.e. tail_entity). This is likely because generating the tail_entity first has a higher chance of making

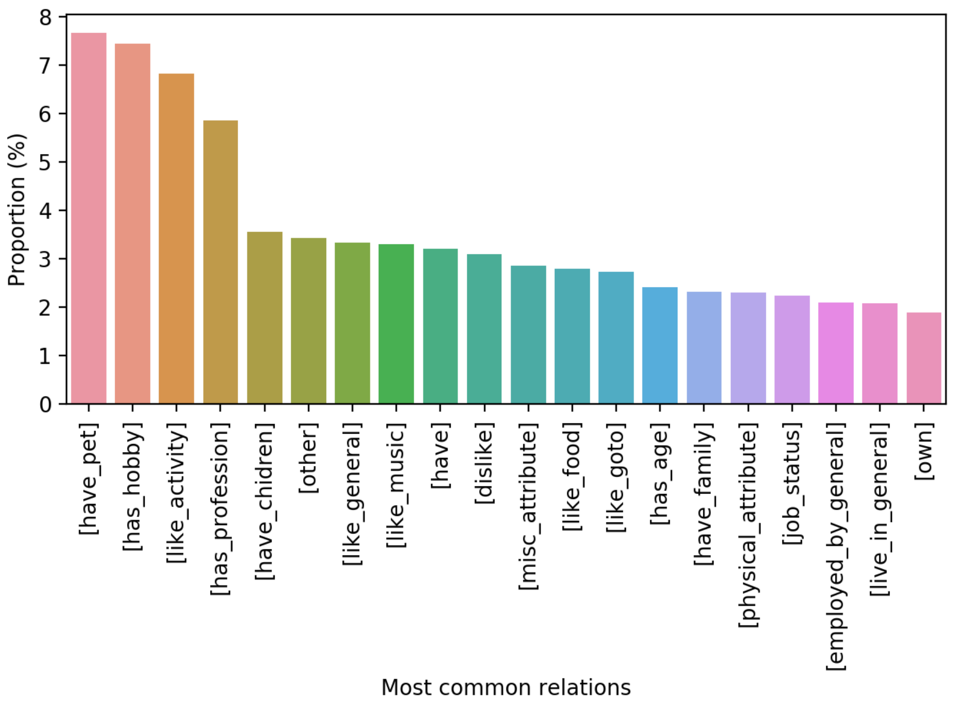

Figure 7: Proportion of most common relations

	all			head_entity			relation			tail_entity		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
head_reln_tail (base)	**38.2**	15.5	18.8	95.4	**90.7**	**92.5**	44.3	**37.1**	**38.4**	55.2	25.1	29.8
head_tail_reln	23.0	7.9	10.1	95.3	63.0	75.4	43.4	10.5	15.4	45.8	18.4	23.1
reln_head_tail	36.7	**15.8**	**19.5**	95.4	81.7	87.6	40.7	34.4	33.5	**56.3**	**28.7**	**34.5**
reln_tail_head	9.8	1.4	2.2	94.0	3.6	6.8	38.9	32.5	31.6	48.0	21.2	26.5
tail_head_reln	2.4	0.7	0.9	95.4	4.8	9.1	29.5	1.7	3.0	41.9	16.7	20.9
tail_reln_head	8.2	1.1	1.7	**96.2**	4.2	8.0	**47.4**	14.0	19.5	51.6	27.3	32.5

Table 6: Performance of unsuccessful models with different training objectives

mistakes early in the generation, which misleads subsequent generation.

8.3.2 Improvement 2a: Optimizing generation hyperparameters

In addition, we tried sampling in the form of top_k (Fan et al., 2018) and top_p (Holtzman et al., 2020) and temperature. For the within-sentence data subset, we also attempted to restrict the generation to only tokens (or token spans) found in the original sentence (as well as special tokens). This was done in two ways. The first is to prevent the model from generating tokens not inside the original sentence. The second is to generate 10 most likely output sequences and retaining the first that contains only tokens/spans from the original sequence. In practice, only the second approach worked reasonably while the first usually terminates early by generate the end-of-text token.

Based on Table 8, using only tokens/spans from the original sentence also improves performance. However, this is because it is implemented by pruning beam search sequences, meaning that its increase in performance can be primarily attributed to beam search. Any form of token sampling did not result in an improvement, likely because relation extraction does not require diversity in the generated output, unlike in natural language (Holtzman et al., 2020).

8.3.3 Improvement 3: In-domain pre-training

Given that the domain of the text is in dialogue, we attempted to use DialoGPT (Zhang et al., 2020)

Hyperparameter	Search range
top_k	1, 5, 10, 20, 50
top_p	0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 0.8, 0.9, 0.95, 0.99
temperature	0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 0.8, 0.9, 0.95, 0.99

Table 7: Generation hyperparameters

	Best	all			head_entity			relation			tail_entity		
	Value	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Base model	-	50.4	24.2	30.3	**97.9**	87.2	92.2	**50.8**	44.3	45.9	68.3	39.4	46.7
Num_beams	10	**51.4**	**26.6**	**32.3**	97.8	**92.7**	**95.1**	49.6	**46.1**	**46.6**	**69.7**	**44.1**	**50.3**
Original_tokens	-	**51.4**	26.4	32.1	97.8	92.6	**95.1**	49.6	**46.1**	**46.6**	69.6	43.9	50.2
Original_spans	-	**51.4**	26.5	32.2	97.8	**92.7**	**95.1**	49.6	**46.1**	**46.6**	**69.7**	44.0	**50.3**
Temperature	0.01	50.4	24.2	30.3	97.9	87.2	92.2	50.8	44.3	45.9	68.3	39.4	46.7
Top_k sampling	1	50.4	24.2	30.3	97.9	87.2	92.2	50.8	44.3	45.9	68.3	39.4	46.7
Top_p sampling	0.01	50.4	24.2	30.3	97.9	87.2	92.2	50.8	44.3	45.9	68.3	39.4	46.7

Table 8: Performance after optimizing generation hyperparameters for within-sentence samples

to initialize the weights of GPT2. DialoGPT was pre-trained using dialogue-like Reddit comment chains.

Pre-training on dialogue corpora (by using DialoGPT) worked less than ideally for our task, as shown in Table 9. This is likely because DialoGPT was pre-trained for multi-turn dialog generation to predict a response given a Reddit comment. On the other hand, our task focuses on extracting relations from a single turn.

	all			head_entity			relation			tail_entity		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Base model	38.2	15.5	18.8	95.4	90.7	92.5	44.3	37.1	38.4	55.2	25.1	29.8
Base model w. DialoGPT	5.4	0.5	0.9	95.9	30.8	46.5	42.5	9.7	14.4	13.6	1.3	2.2

Table 9: Performance of in-domain pretraining