Bootstrapping Relation Extractors using Syntactic Search by Examples

Matan Eyal[1] **Asaf Amrami**[1,2] **Hillel Taub-Tabib**[1] **Yoav Goldberg**[1,2]
[1]Allen Institute for AI, Tel Aviv, Israel
[2]Bar Ilan University, Ramat-Gan, Israel
matane,asafa,hillelt,yoavg@allenai.org

Abstract

The advent of neural-networks in NLP brought with it substantial improvements in supervised relation extraction. However, obtaining a sufficient quantity of training data remains a key challenge. In this work we propose a process for bootstrapping training datasets which can be performed quickly by non-NLP-experts. We take advantage of search engines over syntactic-graphs (Such as Shlain et al. (2020)) which expose a friendly by-example syntax. We use these to obtain positive examples by searching for sentences that are syntactically similar to user input examples. We apply this technique to relations from TACRED and DocRED and show that the resulting models are competitive with models trained on manually annotated data and on data obtained from distant supervision. The models also outperform models trained using NLG data augmentation techniques. Extending the search-based approach with the NLG method further improves the results.

1 Introduction

The goal of Relation Extraction (RE) is to find and classify instances of certain relations in raw text. We denote a binary relation instance, *i.e.* a relation instance with two arguments, with a tuple $\boldsymbol{x} = (\boldsymbol{s}, e^1, e^2, r)$, where $\boldsymbol{s} = [w_0 \cdots w_n]$ is a sequence of sentence tokens, e^1, e^2 are entity mentions within $\boldsymbol{s}$ corresponding to the first and second relation argument, respectively, and $r \in R \cup \{\varnothing\}$ is a relation label from a set of predefined relations of interest, or an indication of 'no-relation'. In binary classification our goal is to classify whether, according to $\boldsymbol{s}$, the entity mentions, e^1 and e^2, satisfy r, the relation label. For such classification we require a training dataset X, comprised of X_p, a set of positive examples, representing the relation of interest, and X_n, a set of negatives examples.

The success of recent papers (Soares et al., 2019; Murty et al., 2020) in supervised RE is fueled by advances in deep learning, but also, crucially, by the availability of a large training set such as TACRED (Zhang et al., 2017), containing tens of thousands of training examples. For most relations of interest, such training data is not available.

In this work we examine methods to inexpensively construct X_p and X_n, in cases where a training set is not available. We are especially interested in constructing the positive set, X_p.

In contrast to common NLP tasks like POS tagging, entity extraction and dependency parsing, the task of relation extraction exhibits a much larger degree of label sparsity. For some relations, even when considering only sentences with entities of the relevant types, the ratio between positive and negative examples is highly skewed toward the latter and obtaining a modest amount of positive examples will require a laborious annotation effort (see §3). While manual annotation of large datasets is a viable approach, it typically requires contracting a team of professional annotators (Doddington et al., 2004; Ellis et al., 2015) or crowd workers (Zhang et al., 2017; Yao et al., 2019) and is not well suited for smaller projects or for ad-hoc extraction tasks.

Our main contribution in this paper is a new methodology built on top of Shlain et al. (2020) for cheaply obtaining large datasets (§6). Shlain et al. (2020) proposed a syntactic search engine that given a lightly annotated example sentence, retrieves new sentences with a similar syntactic structure from a pre-annotated dataset. Our syntactic search bootstrapping method requires a small number of manually curated positive example sentences. Then the search engine matches are used as training data for ML models. We evaluate this approach comparing to human annotated data of varying sizes.

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 1491–1503
April 19 - 23, 2021.

While this method shows promising results with very few user input examples, we also test the impact on performance when more examples are used. One technique for obtaining an abundance of examples uses recent Natural Language Generation (NLG) models (§7.1). It has been shown in recent papers (Wei and Zou, 2019; Anaby-Tavor et al., 2019; Kumar et al., 2020; Amin-Nejad et al., 2020; Russo et al., 2020) that generating abundance of training examples can improve classifier performance. We aim to check whether this can improve our syntactic search method as well.

We evaluate the proposed methodologies by training DL classifiers on the obtained data.

We show that: (1) Syntactic patterns are competitive at bootstrapping training data for ML, even with as little as 3 patterns;
(2) Training DL models over the output of syntactic patterns can significantly improve both recall and F1 over a rule based approach which uses the patterns directly;
(3) Training ML models over the output of syntactic patterns performs better than training models over recently popular NLG data augmentation techniques;
(4) Augmenting the output of syntactic patterns using NLG techniques is often helpful;
(5) Different relations benefit from different strategies.

The code for all our experiments alongside the generation outputs is publicly available[1].

2 Related Work

Distant Supervision. Since its introduction, Distant Supervision (Mintz et al., 2009) has established itself as a viable alternative to manual annotation. Distant Supervision assumes the availability of a knowledge base (KB) of $\langle e^1, r, e^2 \rangle$ triplets where e^1, e^2 are entities known to satisfy relation r. To obtain training examples for a relation r, we sample sentences from a large background corpus: sentences which include entity pairs listed in the KB as satisfying r are labeled positive, the remaining sentences are labeled negative (potentially after satisfying additional constraints). While effective in some cases, the reliance on large pre-existing KBs is a significant limitation. Such KBs are not usually available and the cost of constructing them is high.
Bootstrapping from Rules, Snorkel. To eliminate the reliance on external KBs, Angeli et al. (2015) used the predictions of a rule based extractor on a large corpus to train a first iteration of a statistical extractor. They then continued to refine the extractor through self-training.

Another system which can optionally utilize rules instead of external KBs is Snorkel (Ratner et al., 2017). Snorkel is implementing the data-programming paradigm (Ratner et al., 2016) where ML models are trained in three stages: (i) users write labeling functions that weakly label data points using arbitrary heuristics (*e.g.* extraction rules); (ii) the system learns a re-weighted combination of the labeling functions by explicitly modeling the actual distribution of each class. The results are often precise but low-recall; and (iii) The system uses discriminative models to increase recall while preserving precision.

The techniques used by Angeli et al. (2015) and Snorkel can be effective in increasing the accuracy of the initial labeling rules, but coming up with "good enough" initial rules remains a major challenge. In this sense, the search-based methods suggested in this work for bootstrapping RE datasets are complimentary and can be plugged in as a first step in these multi-step solutions.

Only few papers can be directly compared to our paper and use matches as training-data for ML classifiers. One paper similar in that sense is Angeli et al. (2013) which claims that training a classifier using search-based examples works better than traditional bootstrapping methods. See §6.2 for further compression with Angeli et al. (2013).
Augmentation Through Generation. Similarly to our Example Generation approach, recent papers (Anaby-Tavor et al., 2019; Kumar et al., 2020) suggest using pre-trained language models for data augmentation. In both these papers, the authors suggest prepending class labels to generative models in order to augment the number of instances for classes with a small number of examples. In contrast to these papers we use language models in a zero-shot context, and rather than requiring existing labeled examples of the relevant relation, we propose to manually label the generated samples.

3 The data annotation challenge

In contrast to linguistic annotation tasks such as parts-of-speech, syntactic-trees or semantic roles, annotating data for relation-extraction does not require special expertise. Annotation can be easily

[1] github.com/mataney/BootstrappingRelationExtractors

performed by a motivated native speaker of the language (in case of "every-day" relations such as those available in TACRED and DocRED) or by a domain expert (in case of "specialized" relations such as in biomedicine or law). Annotating a given sentence for a given relation takes roughly the time it takes to read and understand the sentence. So what stops us from obtaining large amounts of annotated data for ML?

The annotation challenge lies in *relation sparsity* in the wild. In an attempt to get a perspective on this issue, let's consider the *founded-by* relation between a PERSON and an ORG, as attested in the TACRED corpus. Assuming we consider only sentences that contain both a person mention and an organization mention, how many sentences do we have to annotate before we reach, for example, 10 positive examples? The TACRED training set has 124 *founded-by* instances, as well as 6947 "negative" instances with matching entity types ("negative" examples are either other relations, or *no-relation*). This 1-out-of-57 ratio indicates that we will likely sample 56 "negative" sentences before hitting a positive instance.[2] This ratio is overly optimistic, as the annotations in the TACRED corpus are already very skewed in favor of positive examples. Even under this very optimistic scenario, we will need to annotate 570 sentences to recover 10 positive examples. The cost of annotation, then, is not in annotating each individual positive sentence, but in finding the sentences to annotate in the first place. Therefore, we should seek for methods that point towards probable positive instances.

In this paper, we present two methods, the first returns close to 1-out-of-1 positive ratio, although with low syntactic diversity, and a second method with roughly 1-out-of-3 positive ratio.

4 Problem Statement and Setup

We are interested in the problem of obtaining a relation classifier for a binary relation, when no a-priori annotated training data for this relation is available. We seek a methodology that will allow to create an effective extractor, using a minimal amount of data annotation effort.

We compare four approaches – manual annotation, syntactic-search, manual annotation over generated examples, and a combination of the last two – to be described in later sections. Here, we discuss setup which is shared to all experiments.

In order to evaluate the methodology on multiple datasets with similar relations, we chose a set of relations that appear in both the TACRED (Zhang et al., 2017) and DocRED (Yao et al., 2019) datasets with at least 50 development examples[3].

To quantify the performance of our methodology we assess it comparing to varying amounts of manually annotated data. In our settings, large amounts of supervised examples represent upper bound for our bootstrapping methods and are not expected.

While relation extraction is often considered as a multi-class classification problem ("find the occurrences of any of these possible relations"), we instead treat the relations separately, training a binary classifier for each one. We believe this is more representative of a user who wishes to target a low number of relations, who is likely to conduct data collection and evaluation for one relation at a time.

Obtaining Negative Examples When training a binary classifier, it is required to include a set of negative examples alongside the list of positive examples. In all our experiments we obtain negative examples by looking for sentences that contain entity types that are compatible with the relation (i.e, for the *founded-by* relation we sample sentences that include both a PERSON and an ORG). In our syntactic based methods we sample from the same domain as our positive examples (Wikipedia) and then filter this list by removing sentences in which the entities are connected by a syntactic pattern which is attested by the positive examples. For the supervised baselines of various sizes, we obtain negative examples by sampling them from the annotated training set, without replacement.[4]

Datasets We used two datasets to explore our different methods. TACRED (Zhang et al., 2017), a large-scale multi-class relation extraction dataset built over newswire and web text. And DocRED (Yao et al., 2019), a dataset for document level RE, and similarly designed for multi-class prediction. Per our setup above, we changed the setting of both datasets to per relation binary classi-

[2] See Appendix A for similar distributions over all relations.

[3] *org:country of headquarters*, *org:founded by*, *per:children*, *per:city of death*, *per:date of death*, *per:origin*, *per:religion*, *per:spouse* for TACRED, and similarly *headquarters location*, *founded by*, *child*, *place of death*, *date of death*, *country of origin*, *religion*, *spouse* for DocRED.

[4] The **positive to negative** ratio in training data has an effect on the resulting model's quality. We experimented with positive-to-negative ratios of 1, 5, 10 and 20, as well as with a "match the dev-set" ratio. We found a ratio of 10 negative examples for each positive sentence to performs well.

fication. As our main goal in this paper is to evaluate different bootstrapping methods, and not novel methods for document-level relation extraction, we chose to include only instances with single supporting sentence in DocRED (*i.e.* sentence level relations). As DocRED's labelled test set is not publicly available, we used the development set as our test set and used 20% of the train set as development set.

Models Our classifiers throughout the following experiments are based on the Entity Markers architecture (Soares et al., 2019). In the paper, the authors proposed wrapping the relation arguments with marker tokens (*e.g.* $[E1_{start}]$ John $[E1_{end}]$ was born in $[E2_{start}]$ 1948 $[E2_{end}]$). The altered text is then passed as input to a BERT model (Devlin et al., 2018) where the relation between the two entities is represented by the concatenation of the final hidden states corresponding to their respective start tokens. Finally, this representation is fed into a classification head and the model is fine-tuned for relation classification. *cf.* (Soares et al., 2019) for more details. We use a similar model with the exception that we use a more recent pretrained language model, RoBERTa (Liu et al., 2019), and perform binary, rather than multi-class, classification.

In all of the following experiments we trained our model with 3 different random seeds to lower variance introduced to the model with different initializations, and report the average score. At inference time we set the prediction threshold value for the test set to be the cut-off value that maximized F1 over the development set.

5 Manual Annotation Baseline

Setup. Our comparison point throughout the paper is a model trained on traditionally-collected annotated data. We sample increasing-sized annotated sets from TACRED and DocRED, containing 55, 110, 220, 550, and 1100 examples. These correspond to 5, 10, 20, 50, 100 positive examples with 50, 100, 200, 500, 1,000 negatives examples. Additionally, we measure the performance on these datasets when using all available positive examples for each relation.

Results Listed in the top rows of Table 1, averaged over all relations. Unsurprisingly, increasing the number of examples increases performance, with the exception of DocRED on which using all positive labels performs slightly worse than using 100 sampled positive examples for each relation, we attribute this to sampling noise. DocRED scores are generally lower than TACRED scores. This is because of the way we constructed the development and test sets: while in TACRED's development set each sentence includes a single entity pair with a single relation, in DocRED, we pass all possible sentences with entity pairs of the same type as the evaluated relation as possible candidates. This dramatically increases the number of candidates, and by that of possible type I errors. Moreover, as we included only examples with exactly one supporting sentence, the number of positive examples is low for some of the relations. All of this effects DocRED classification scores comparing to TACRED.

Importantly, in all these experiments, the number of annotated examples used is significantly higher than the number used in our Syntactic Search experiments (3 examples in total).

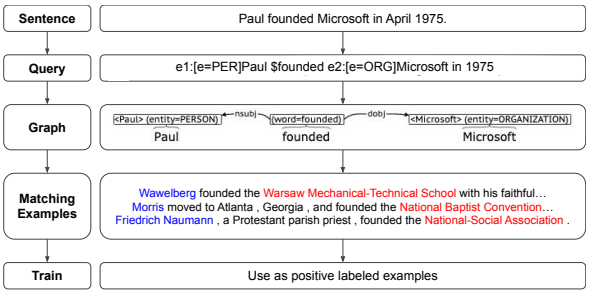

Figure 1: Flow of the Syntactic Search by Example method. For details, see §6.

6 Syntactic Search by Example

We consider this section to be the main contribution of the work. We show that:
(i) with modern DL modeling, effective relation extractors can be trained using sentences derived from less than a handful of syntactic patterns; and (ii) through the use of by-example syntactic search engines, one can construct these patterns very quickly, without needing to understand syntax.

To explain the suggested workflow, let's consider a user who wants to train a relation extraction binary classifier for the *founded by* relation, and has a single example sentence, "Paul founded Microsoft in April 1975". Patterns over syntactic structures and entity types are very effective for deriving high-precision extraction templates. For example, searching for sentences containing the word "founded" with an *nsubj* dependency of type PERSON and *dobj* dependency of type ORG, will return many matches for the *founded-by* relations.

Method	TACRED	DocRED
Annotated 5+50	0.097	0.140
Annotated 10+100	0.136	0.215
Annotated 20+200	0.266	0.271
Annotated 50+500	0.458	0.311
Annotated 100+1000	0.516	0.321
Annotated All	0.569	0.306
Pattern Based RE (3 qrs)	0.128	-
Synt. Search (3 queries)	0.443	0.266
Example Generation	0.439	0.109
Search + Generation	0.491	0.277

Table 1: Average test F1 score over all relations. Pattern Based RE was given 3 positive patterns. Synt. Search is trained on data created from same 3 patterns. The Annotated experiments are denoted by the number of positive examples + negative examples.

There are two issues with this approach (1) while high-precision, the recall of the patterns is low; and (2) syntactic patterns require both linguistic and computational expertise to specify and execute.

The premise of this paper is that the low recall can be offset by machine learning. The sentences resulting from syntactic search over a few patterns are diverse enough that an ML model trained over them manages to generalize from the specific syntactic pattern and identify a broader range of cases, increasing recall substantially. We show this is indeed the case.

To overcome the need for linguistic expertise we propose using a by-example syntactic search engine (Shlain et al., 2020)[5] which allows users to execute syntactic queries based on example sentences: the user enters a sentence satisfying the relation of interest and annotates it with light markup indicating the arguments and the trigger words. The system then automatically translates the markup into a syntactic pattern, matches it against a large pre-annotated corpus (*e.g.* all Wikipedia sentences), and returns results. The user does not need to be familiar with syntactic formalisms or with advanced NLP.

6.1 By-example Patterns for Collecting Training Data

Fig. 1 demonstrates the user process. Starting with the sentence *Paul founded Microsoft in April 1975*, the user marks *Paul* as *e1* (`e1:`) with an entity-type restriction of PERSON (`[e=PER]`), *Microsoft* as *e2* (`e2:`) with an entity-type ORG (`[e=ORG]`), and *founded* as a trigger word (`$founded`). The SPIKE system translates the query into a syntactic graph, which is then matched against Wikipedia, returning 11,345 sentences matching the pattern (note that the word 'founded' is matched lexically, while Paul and Microsoft become place holders for any person and any organization that adhere to the syntactic configuration). A subset of the returned sentences is then used as positive examples for model training.

While 11,345 cases make an impressive training set, these sentences share the same core syntactic configuration, and classifiers, trained on these matches, will not necessarily generalize well. The matches will also share the exact same lexical predicate ("founded"). The lack of lexical diversity of the predicate can be expanded by the user by supplying alternative words, perhaps aided by distributional similarity methods such as word2vec, or by querying a bi-LM such as BERT (Devlin et al., 2018) (§6.2.1). To counter the lack of structural diversity the user can supply additional patterns, derived from example sentences. For example, the user may supply also '[$_{e_2}$Microsoft]'s founder [$_{e_1}$Paul]' (possessive construction) and '[$_{e_2}$Microsoft] was founded by [$_{e_1}$Paul]' (passive) as additional patterns (§6.2).

6.2 Experiments and Results

Setup For each relation, we select 3 representative sentences and annotate them based on the process described above[6]. We do not perform any lexical expansion of trigger words beyond the initial pattern at this point. The queries are processed by SPIKE (Shlain et al., 2020) and the results are used as positive instances in the generated training set. A full list of the SPIKE queries we used can be found in appendix D, Table 5.

We also compare the TACRED classifier to a rule based extractor which uses the syntactic queries directly. Each syntactic query is added as a syntactic pattern to this extractor: any sentence which satisfies one of the syntactic patterns is labeled as a positive instance; sentences which do not satisfy any of the patterns are labeled negative.

Results Listed in the *Synt. Search* and *Pattern*

[5] https://spike.apps.allenai.org

[6] In this experiment, the selection of representative sentences is based on a heuristic process: we intuitively conceive of basic sentences exemplifying the relation, construct the corresponding Spike queries and briefly validate the number and quality of the returned results. We limit the number of seed examples to 3 since we believe coming up with 3 examples should be simple even for non-experts. In §7.1 we show that using more seed examples can further improve performance.

Dataset	Predicates	100	500	1000
TACRED	One Trig.	0.487	0.459	0.461
	Trig. List	0.517	0.490	0.478
DocRED	One Trig.	0.290	0.336	0.338
	Trig. List	0.316	0.338	0.337

Table 2: F1 scores for *founded by*, *child*, *place of death* and *date of death* and *spouse* when expanding the triggers list for the Syntactic Search "by Example" method.

Based RE rows of Table 1[7], *Pattern Based RE*, using just the 3 patterns per relation, achieves a very low F-score of 12.8%, due to low recall. However, this is already competitive with training a classifier on 5-10 positive examples per relation. Training a classifier on the extracted relations increases the scores significantly, to 44.3F_1 on TACRED and 26.6F_1 on DocRED, approaching supervised training on 50+500 annotations (for TACRED) or 20+200 annotations (for DocRED). This result demonstrates that training an ML model over the output of a rule based model can significantly improve performance, echoing similar conclusions in Angeli et al. (2013). Interestingly, Angeli et al. (2013) used a total of 4,697 patterns across 41 relations, an average of 114 patterns per relation. We demonstrate that by applying syntactic patterns to a large corpus and using modern DL classifiers, results competitive with manual annotation baselines can be reached with as few as 3 syntactic rules.

6.2.1 Syntactic Search with Trigger Expansion

Setup Constructing queries from 3 seed sentences produces retrieved sentences with low lexical diversity. *e.g.* if all the seed sentences for *founded-by* use the word *"founded"* to express the relation, then all retrieved sentences will likewise include the word *"founded"*, and exclude alternatives like "established", "formed", "started", etc.

In this experiment we generalize the seed queries to allow a list of trigger words rather than a single word. We consider only relations which include a lexical trigger in their seed patterns[8]. Alternative triggers are selected by reviewing the closest words to the original triggers in word2vec's embedding space (Mikolov et al., 2013). Appendix C includes the lists of alternative lexical triggers used. We train classifiers on 100+1000, 500+5,000 and 1,000+10,000 examples obtained from these expanded-trigger queries.

Results As illustrated in Table 2, adding alternative triggers improves results across all sample sizes for TACRED and for the 100+1000 size in DocRED.

7 Augmenting Syntactic patterns with Natural Language Generation

We showed how the Syntactic Search by Example method works with only a few human annotated examples. In this section we would like to pursue NLG based methods to expand the number of exemplary patterns. Generative language models, compared to other methods for data augmentation (*e.g.* Iterative bootstrapping and distant supervision) are highly accessible and require low technical expertise (sometimes passing a prompt is enough). Moreover, recent papers (Wei and Zou, 2019; Anaby-Tavor et al., 2019; Kumar et al., 2020; Amin-Nejad et al., 2020; Russo et al., 2020) report high impact of such models for the closely related Data Augmentation task. We therefore present numerous methods that take advantage of such models for RE bootstrapping.

First we show how a user can produce a high number of generated sentences using GPT2 (Radford et al., 2019). Then we demonstrate how the generated sentences can be integrated in the Syntactic Search by Example method (§7.1). Finally, in order to validate the necessity of the syntactic search in this flow we compare it to feeding the raw generations as inputs to a classifier (§7.2).

Generating Examples LM

The user-flow Depicted in Fig. 2: The user enters a relation prompt ("Paul founded Microsoft"), to which the system responds by returning sentences that express the same relation. While not all returned sentences express the relation, many of them do. To filter out out-of-relation sentences the user goes through the list until she identifies a predefined number of positive examples (Here we used 100 sentences). In our experiments we encountered 1 positive example for every 3 examples annotated. This 1-out-of-3 ratio is significantly better than blindly sampling from a corpus (1-out-of-57, see §3), and by that can considerably save annotation time. For each example, the user marks the relevant entities, and optionally also the trig-

[7] Results correspond to 100+1,000 (TACRED) and 1,000+10,000 (DocRED) examples, for results and discussions of different dataset sizes, see Appendix B.

[8] *per:children*, *per:date of death*, *org:founded by*, *per:city of death* and *per:spouse*, and DocRED's *child*, *date of death*, *founded by*, *place of death* and *spouse*

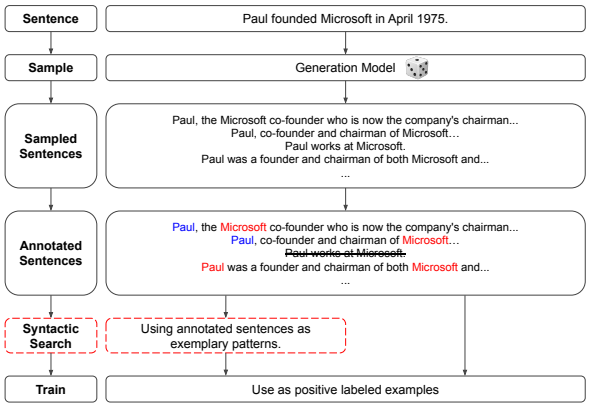

Figure 2: Flow of sampling examples from conditional language model. The "Syntactic Search" step corresponds to §7.1, while skipping this step, corresponds to §7.2.

ger word (the main word indicating the relation). These examples are then used as additional input examples to the syntactic search engine (§7.1) or as train datasets for ML models (§7.2).

Technical details We begin with a large pre-trained LM (we use GPT2-medium (Radford et al., 2019)), and fine-tune it to the generation task. The method assumes the availability of relation-annotated data, though its relations do not need to overlap with the ones we are attempting to extract (in our case, we ensured the groups are distinct). The approach can be considered as an instance of transfer-learning, where we attempt to transfer the example-generation knowledge from the training relations to novel relations. Given the annotated RE dataset, we consider positive examples of the form $(\boldsymbol{s}, e_1, e_2, r)$, where $r \in R$. We transform each instance to a conditioned LM training example, in which the LM sees a prefix (prompt) and should complete it. In our case the prompt is derived from (e_1, e_2, r), followed by a special symbol, and we train the LM to produce the corresponding sentence $\boldsymbol{s}$. To derive the prefix we apply a pre-defined *template* associated with each relation r[9]. The template has two slots to be filled with the entities e_1 and e_2. For example, a template for the *founded-by* relation can take the form $[e_2]$ *founded* $[e_1]$. We then fine-tune GPT2 on these training examples. At inference time, the user provides a *single* prompt based on their desired relation.

Given the user prompt, we generate 1000 sentences with nucleus sampling (Holtzman et al., 2019) of 0.99 and length of up to 50 tokens. We annotate the generated sentences until reaching 100 positive instances (usually requiring 200-300 sentences), this takes up to 1.5 hours per relation. These generated sentences are annotated and used as inputs to the syntactic search method (§7.1) or directly as positive examples to a classifier (§7.2).

[9]In our experiments, we use on average 3 different templates for each relation type, so a single annotated relation example will result in 3 (on average) different fine-tuning examples for the LM, each with a different prompt.

7.1 Enhancing Syntactic Search with Example Generation

We integrate the generation outputs in the Syntactic Search by Example method by taking the positive annotated examples (on which we mark the entities as part of the annotation process) and automatically transforming them into SPIKE queries. This step has the potential to add substantial syntactic and lexical diversity to the pattern set, resulting in both larger and more diverse sets of positive examples. This combines the best of both worlds: the generative model is used to provide structural and lexical diversity, while the syntactic search system is used to provide a large selection of naturally occurring corpus sentences adhering to these patterns.

Experiments and Results

Setup To reduce noise, we exclude queries where more than 1 out of 5 sampled results does not express the relation of interest. On average, we increased the number of syntactic patterns to 9.25, ranging from 6 to 14 after filtering.

Results As listed in the *Search + Generation* row of Table 1, this method achieved best performance for both TACRED and DocRED with overall scores corresponding to 550/1100 and 220/550 annotated examples respectively. Using the generation outputs as examples doesn't only help in suggesting more sentences satisfying the relation but also in augmenting the number of predicates used. We looked on the number of predicates used for the TACRED relations which include lexical triggers (similarly to §6.2.1), the generation phase suggested 7.4 predicates on average, more than the 2.8 predicates per relation of our original patterns, and less than the trigger expansion method we suggested in §6.2.1, where we tried to find all the possible predicates, with 18.2 triggers on average. We conclude that while the Syntactic Search by Example method performs well with only a few example patterns, this can be even improved with more input examples. While we report Syntactic Search by Example enjoys such generation-based pattern aug-

mentation, a similar boost with different, non-NLG, methods is of course possible. We leave further probing for other pattern augmentation methods as future work.

7.2 Directly Training Classifiers using Generation Outputs

It is possible that generative models produce diverse enough training examples that will suggest our syntactic search superfluous. We validate the necessity of taking the annotated generations (Annotated similarly to §7.1) through the Syntactic Search by Example method, by comparing it to simply passing the annotated generations as classifier inputs, as depicted in the RHS of Fig. 2.

Experiments and Results

Setup Many of the samples include the entities from the prompt verbatim. Before using them as the model inputs, we replace the entities with a random Wikipedia entity of the same type.

Results As can be seen in the *Example Generation* row in Table 1, on TACRED, this method produces F1 scores on par with Syntactic Search by Example. However, evaluating on DocRED, the method does not produce competitive results[10]. On both datasets it produce worse than *Search + Generation*. We conclude that it is more beneficial to use outputs of generative models as syntactic search queries, and by that find syntactically similar sentences, comparing to simply use generations as the train set. We deduce models are likely to generalize better on "real world" examples.

8 Additional Experiments

8.1 Results across relations

Analyzing the results we highlight some interesting trends (Fig. 3). First, we note that the behavior is not consistent between relations, nor datasets: different relations behave differently, showing different trade-offs between different methods.

Classifiers for relations like *"Religion"*[11], *"City of Death"* and *"Date of Death"* seem to plateau at around 50-100 manually annotated examples. For these relations, annotating more data is not necessarily useful. The syntactic search approach works especially well for these relations: applying syntactic search over 3 seed queries is sufficient to yield results on par or slightly higher than all available manually annotated data. We hypothesize that these findings might be the result of low diversity in the ways these relations are typically expressed.

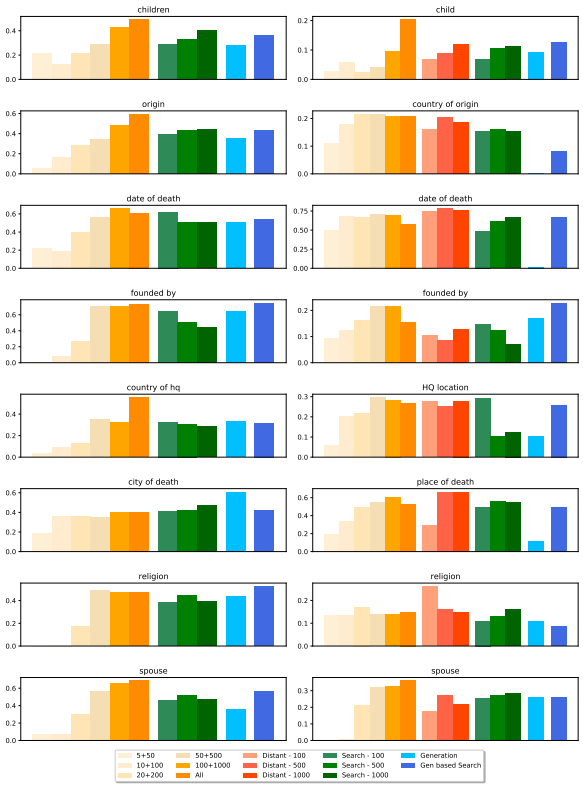

Figure 3: F1 scores of TACRED (right), and DocRED (left) by relation.

While the combined *Search + Generation* approach is overall useful, the effect is not consistent across relations: performance improves for some relations and deteriorates for others. In §7.1 we described the techniques we use to reduce the noise coming from additional queries. These techniques however are rather basic and these results indicate that more advanced techniques of the type used in Angeli et al. (2015) and Ratner et al. (2017), are likely to yield more consistent improvements.

8.2 Distant Supervision

Setup Distant supervision (Mintz et al., 2009) suggests a method to construct a training dataset based on a large external KB of relation triplets. Yao et al. (2019) offered a machine annotated version of DocRED constructed by aligning Wikipedia pages

[10]The language model used to generate examples was fine-tuned on a version of TACRED which excludes the relations we evaluate on. Still, for TACRED, the language model is fine-tuned and evaluated on data from the same domain (newswire). The DocRED data on the other hand, is taken from Wikipedia, so the evaluation is essentially out of domain. We therefore conclude that used independently, this approach is applicable only in cases where a background RE dataset is of the same domain as the target corpus from which we want to extract relations.

[11]TACRED's "religion" relation plateaus as it has a low number of train instances.

with Wikidata. The authors took great care in creating this resource: a high-quality NER model trained on in-domain manually annotated data was used to automatically annotate possible relation arguments; a named entity linker was used to merge entities with similar KB ID; and finally, Wikidata was queried in order to label pairs of linked entities.

We trained a classifier using the released data, sampling increasing number of examples: (100+1,000, 500+5,000, 1,000+10,000). We report best score of $0.312 F_1$ (500+5,000 split).

Results This Distant Supervision dataset, created by Yao et al. (2019), appears to be of very high quality and the results are on par with the full set of manually annotated data. These results indicate that given a large KB of relation triplets, a high-quality in-domain NER, and a high quality linking solution, distant-supervision is a very promising technique. It should be noted however, that the availability of all these external resources is very rare in practice and is not required by the methods proposed in this work.

9 Applicability to other languages

We explored only English in this work. However, we argue that our main method – example-based syntactic search followed by DL-training – is not strongly tied to English, and we encourage other researchers to experiment with it in their languages of interest. We provide details of what is needed to adapt the system to a different language.

The Syntactic Search by Example method requires (1) An automatically dependency-parsed corpora in the language. These can be readily produced by the many syntactic parsers that are available for many languages (Manning et al., 2014; Honnibal and Montani, 2017; Qi et al., 2020). (2) An indexing engine that supports efficient queries over parse trees. Shlain et al. (2020) uses the open-source Odinson engine (Valenzuela-Escárcega et al., 2015) for this purpose. (3) A component that translates a query in spike's "by example" syntax to the indexing engine's query syntax. This requires finding the minimal (in terms of number of nodes) sub-graph that connects all relation arguments (and predicates if available), then search for sentences with similar sub-graphs in the index. With these three components, a syntactic-search system can be readily implemented. The rest of the components are straightforward application of DL methods. Indeed, we suspect the major obstacle in application to a new language will be the availability of evaluation data.

10 Conclusion

We show that with modern DL classifiers and a dataset bootstrapped using syntactic search with as few as 3 seed patterns can be as effective as a dataset with hundreds of manually annotated samples. Using LMs help to further diversify the dataset and improve results. Overall, our results are positively optimistic for bootstrapping methods. However, this work is only an initial step in exploring methods for bootstrapping relation extractors using minimal user effort, supported by strong pre-trained neural LMs. We hope to encourage further work in this direction.

11 Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 802774 (iEXTRACT).

References

Ali Amin-Nejad, Julia Ive, and Sumithra Velupillai. 2020. Exploring transformer text generation for medical dataset augmentation. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4699–4708.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2019. Not enough data? deep learning to the rescue! *arXiv preprint arXiv:1911.03118*.

Gabor Angeli, Arun Tejasvi Chaganty, Angel X Chang, Kevin Reschke, Julie Tibshirani, Jean Wu, Osbert Bastani, Keith Siilats, and Christopher D Manning. 2013. Stanford's 2013 kbp system. In *TAC*.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Tejasvi Chaganty, Jason Bolton, Melvin Jose Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D Manning. 2015. Bootstrapped self training for knowledge base population. In *TAC*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, page 1. Lisbon.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie M Strassel. 2015. Overview of linguistic resources for the tac kbp 2015 evaluations: Methodologies and results. In *TAC*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Shikhar Murty, Pang Wei Koh, and Percy Liang. 2020. Expbert: Representation engineering with natural language explanations. *arXiv preprint arXiv:2005.01932*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, 11 3:269–282.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575.

Giuseppe Russo, Nora Hollenstein, Claudiu Musat, and Ce Zhang. 2020. Control, generate, augment: A scalable framework for multi-attribute text generation. *arXiv preprint arXiv:2004.14983*.

Micah Shlain, Hillel Taub-Tabib, Shoval Sadde, and Yoav Goldberg. 2020. Syntactic search by example. In *Proceedings of ACL 2020, System Demonstrations*.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. *arXiv preprint arXiv:1906.03158*.

Marco A Valenzuela-Escárcega, Gus Hahn-Powell, Mihai Surdeanu, and Thomas Hicks. 2015. A domain-independent rule-based framework for event extraction. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 127–132.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. Docred: A large-scale document-level relation extraction dataset. *arXiv preprint arXiv:1906.06127*.

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.

A Estimated effort for Annotation

Table 3 lists for each relation the ratio of positive to negative examples in the TACRED training set. A negative example for a relation *r*, is any example whose entities share types with a positive example of *r*, but whose label is different from *r*. Note that TACRED significantly under represents negative examples so the reported ratio is an upper bound on the ratio in the wild.

Relation	Pos/Neg Ratio
org:country_of_hq	1/7
org:founded_by	1/56
per:children	1/64
per:city_of_death	1/31
per:date_of_death	1/26
per:origin	1/10
per:religion	1/2
per:spouse	1/52

Table 3: Pos/Neg ratio in TACRED, rounded to the closest fraction.

B Syntactic Search by Example with varying dataset sizes

We experimented with varying the number of sampled examples, using the same 3 seed syntactic patterns. The results are reported in Table 4. While DocRED's F1 scores increase with increasing number of sampled examples, the trend is opposite in TACRED. We believe this is due to different initializations and inductive noise in both the positive and negative samples introduced by sampling from semi-noisy data.

Method	**TACRED**	**DocRED**
Synt. Search - 100	0.443	0.250
Synt. Search - 500	0.434	0.259
Synt. Search - 1000	0.427	0.266

Table 4: Syntactic Search by Example with different training sizes

C Trigger List Expansion

For the majority of patterns used in the Syntactic Search by Example experiments we used a single trigger word (see Appendix D). To experiment with using trigger lists, we modified the patterns in Appendix D in the following way:

We changed the triggers in all *child\children* patterns to include any of the following possibilities:

baby, child, children, daughter, daughters, son, sons, step-daughter, step-son, step-child, step-children, stepchildren, stepdaughter, stepson

For *founded-by* relations we change the "founder" trigger to be any of these triggers:

founder, co-founder, cofounder, creator

and changed "founded" to be any trigger from the following list:

create, creates, created, creating, creation, co-founded, co-found, debut, emerge, emerges, emerged, emerging, establish, established, establishing, establishes, establishment, forge, forges, forged, forging, forms, formed, forming, founds, found, founded, founding, launched, launches, launching, opened, opens, opening, shapes, shaped, shaping, start, started, starting, starts

In *spouse* relations we expanded the "husband\wife" trigger to be any of:

ex-husband, ex-wife, husband, widow, widower, wife, sweetheart, bride

and the "marry" trigger to:

divorce, divorced, married, marry, wed, divorcing

For the *"date of death"* and *"place\city of death"* we changed the "died" trigger to any of:

died, executed, killed, dies, perished, succumbed, passed, murdered, suicide

D Examples used for Syntactic Search by Example

child
<>e1:[e=PER]John 's t:[w={triggers}]daughter , <>e2:[e=PER]Tim, likes swimming.
<>e1:[e=PER]Mary did something to her t:[w={triggers}]son, <>e2:[e=PER]John in 1992.
<>e1:[e=PER]Mary was survived by her 4 t:[w={triggers}]sons, John, John, <>e2:[e=PER]John and John.
triggers = son | daughter | child | children | daughters | sons
founded by
<>e1:[e=ORG]Microsoft t:[w]founder <>e2:[e=PER]Mary likes running.
<>e2:[e=PER]Mary t:[w]founded <>e1:[e=ORG]Microsoft.
<>e1:[e=ORG]Microsoft was t:[w]founded $by <>e2:[e=PER]Mary.
headquarters location
John Doe, a professor at the <>e1:[e=ORG]Oxford <>in:[t=IN]in <>e2:[e=LOC]England likes running.
<>e1:[e=ORG]Oxford, a leading <>t:[t=NN]company <>in:[t=IN]in <>e2:[e=LOC]England.
<>e2:[e=LOC]England pos:[t=POS]'s largest university is <>e1:[e=ORG]Oxford.
religion
<>e1:[e=PER]John is a e2:[w={triggers}]Jewish,,
e2:[w={triggers}]Jewish <>e1:[e=PER]John is walking down the street.
<>e1:[e=PER]John is a e2:[w={triggers}]Methodist Person.
triggers = Methodist | Episcopal | separatist | Jew | Christian | Sunni | evangelical | atheism | Islamic | secular | fundamentalist | Christianist | Jewish | Anglican | Catholic | orthodox | Scientology | Islamist | Islam | Muslim | Shia
spouse
<>e1:[e=PER]John 's t:[w=wife | husband]wife, <>e2:[e=PER]Mary , died in 1991.
<>e1:[e=PER]John t:[l]married <>e2:[e=PER]Mary,,
<>e1:[e=PER]John is t:[w]married to <>e2:[e=PER]Mary,
origin
<>e2:[e=MISC]Scottish <>e1:[e=PER]Mary is high.
<>e1:[e=PER]Mary is a <>e2:[e=MISC]Scottish professor.
<>e1:[e=PER]Mary, the <>e2:[e=LOC]US professor.
date of death
<>e1:[e=PER]John was announced t:[w]dead in <>e2:[e=DATE]1943.
<>e1:[e=PER]John t:[w]died in <>e2:[e=DATE]1943.
<>e1:[e=PER]John, an NLP scientist, t:[w]died <>e2:[e=DATE]1943.
place of death
<>e1:[e=PER]John t:[w]died in <>e2:[e=LOC]London, <>country:e=LOC England in 1997.
<>e1:[e=PER]John t:[w]died in <>e2:[e=LOC]London in 1997.
<>e1:[e=PER]John $-LRB- t:[w]died in <>e2:[e=LOC]London $-RRB-.
DocRED's *founded by*
<>e1:[e=ORG]MISC Microsoft t:[w]founder <>e2:[e=PER]Mary likes running.
<>e2:[e=PER]Mary t:[w]founded <>e1:[e=ORG]MISC Microsoft.
<>e1:[e=ORG]MISC Microsoft was t:[w]founded $by <>e2:[e=PER]Mary.
DocRED's *origin*
<>e2:[e=MISC]Scottish company, <>e1:[e=ORG]Microsoft is successful.
<>e1:[e=ORG]MISC Microsoft is a <>e2:[e=MISC]Scottish Company.

Continued on next page

<>e1:[e=ORG]MISC Microsoft is a <>t:[t=NN]song $by <>e2:[e=MISC]Scottish musician.
DocRED's *date of death*
<>e1:[e=PER]John $-LRB-
<>e1:[e=PER]John t:[w]died in <>e2:[e=DATE]1943.
<>e1:[e=PER]John, an NLP scientist, t:[w]died <>e2:[e=DATE]1943.
DocRED's *place of death*
<>e1:[e=PER]John t:[w]died in <>e2:[e=LOC]London, <>country:e=LOC England in 1997.
<>e1:[e=PER]John t:[w]died in <>e2:[e=LOC]London in 1997.
<>e1:[e=PER]John $-LRB- $[e=DATE]1997, $[e=LOC]London $- $[e=DATE]1997 <>e2:[e=LOC]London $-RRB-.
DocRED's *headquarters location*
<>e1:[e=ORG]Microsoft, a leading <>t:[t=NN] company <>in:[t=IN]in <>e2:[e=LOC]Redmond.
<>e1:[e=ORG]Microsoft is t:[l=base | headquarter]based in <>e2:[e=LOC]England.
<>e1:[e=ORG]Microsoft, a leading <>t:[t=NN] company based <>in:[t=IN]in <>e2:[e=LOC]Redmond.

Table 5: SPIKE search patterns for TACRED and DocRED relations.