Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5569–5578
Marseille, 11–16 May 2020

Reproducing Neural Ensemble Classifier for Semantic Relation Extraction in Scientific Papers

Kyeongmin Rim, Jingxuan Tu, Kelley Lynch, James Pustejovsky
Brandeis University, Department of Computer Science
{krim,jxtu,kmlynch,jamesp}@brandeis.edu

Abstract

Within the natural language processing (NLP) community, shared tasks play an important role. They define common goals and allow the comparison of different methods on the shared problems and data. SemEval-2018 Task 7: Semantic Relation Extraction and Classification in Scientific Papers (SE18T7) involves the identification and classification of relations in abstracts from computational linguistics (CL) publications. In this paper we describe an attempt to reproduce the methods and results from the top performing system at for SE18T7. We describe challenges we encountered in the process, report on the results of our system, and discuss the ways that our attempt at reproduction can inform best practices.

Keywords: REPROLANG, Reproducibility, Semantic Relation Extraction, Scientific Information Extraction

1. Introduction

Replicability and reproducibility are core ideas of modern scientific methods (Ivie and Thain, 2018), and thus reproducibility of scientific research results is an important topic in many research areas. Particularly over the last decade, terms such as "reproducibility crisis" or "p-hacking" have gained attention from different scientific communities, especially after studies showed failure in reproduction of researches in life and medicine science (Begley and Ellis, 2012), psychology ans behavioral science (Anderson et al., 2015), as well as astrophysics (Collaboration and others, 2015). The same studies also showed that a complete and successful replication of experiments and results in some published articles with huge findings and impact is extremely difficult or nearly impossible because of the lack of model details and experiment settings .

The field of natural language processing (NLP) and computational linguistics (CL) hasn't been an exception. The questions surrounding reproducibility of published work have been getting researchers attention over the last decade. (Fokkens et al., 2013; Hagen et al., 2015). More recently, *Language Resource and Evaluation* initiated a proposal to establish a venue for publications on the topic of replicability and reproducibility to encourage researchers to actively investigate existing methods and experiments, and discuss the positive or negative results (Branco et al., 2017). Workshops dedicated to these topics include

- *RRNLP 2015 - Workshop on Replicability and Reusability in Natural Language Processing*[1]
- *4REAL Workshop - Workshop on Research Results Reproducibility and Resources Citation in Science and Technology of Language*[2] (Branco et al., 2016)
- *4REAL 2018 - Workshop on Research Results Reproducibility and Resources Citation in Science and Technology of Language*[3] (Branco et al., 2018)

, that aimed to address the need for replication of published results and push the discussion of this topic much further. A recent study (Mieskes et al., 2019) showed, by conducting a survey, the significance of reproducibility to a scientific community as well as the importance of having a common model to share replicable research artifacts. Most recently, the Shared Task on the Reproduction of Research Results in Science and Technology of Language (REPROLANG 2020)[4] is dedicated to motivating the spread of scientific work on reproduction as a type of a collaborative shared task. The selected tasks for REPROLANG are about reproducing results of a set of published articles that focus on different language technologies, including lexical processing, sentence processing, text processing, applications and language resources. Our work targets Task C.1 of REPROLANG 2020, *Relation extraction and classification*.

In this paper, we aim to reproduce an automatic scientific relation extraction and classification system (henceforth ETH-DS3Lab system) described in Rotsztejn et al. (2018) (henceforth RHZ). The ETH-DS3Lab system was originally submitted to SemEval-2018 Task 7: Semantic Relation Extraction and Classification in Scientific Papers (SE18T7) (Gábor et al., 2018) and the reported results ranked first place for 3 subtasks out of 4. Throughout this paper, we will describe our steps of re-implementing the system and replicating the experiments and results in the most detailed manner. In doing so, we will also show difficulties we faced partly due to the nature of the task itself, and partly due to the under-specified or missing details in the RHZ And finally, we'll end with our suggestions to the community for better practice in writing and reporting statistically driven CL researches for reproducibility and transparency.

The rest of the paper is structured as follows. In the next section (sec 2.) we review some related work and motivation for this type of relation extraction and classification task. Then section 3. fills in the technicality on how SE18T7 formalized and how dataset look like. In section 4., we describe in detail the ETH-DS3Lab system we replicated. Then in following section 5., the replicated system and challenges we faced as well as adjustments we made are described at component-level. In section 6., we present the results from our replica system and draw comparison

[1] `http://nl.ijs.si/rrnlp2015/`
[2] `http://4real.di.fc.ul.pt/`
[3] `http://4real2018.di.fc.ul.pt/`
[4] `https://lrec2020.lrec-conf.org/en/reprolang2020/`

with the results from the ETH-DS3Lab system. In section 7., we continue discussion on reproducibility of the ETH-DS3Lab system and our suggestions for more reproducible scientific reporting. And finally in section 8., we end with a conclusion of this work. Our replica system is available on a public code repository[5]

2. Prior study: Relation Extraction and Classification

As the academic community evolves and thrives, scientific literature and publications are also growing at an unprecedented rate (Johnson et al., 2018). Thus easy accessibility and efficient retrieval of in-domain scientific knowledge are becoming one of the most benefiting goals in all scientific and academic fields. Consequently, with the advancement of Information Retrieval (IR) techniques and Human Language Technology (HLT), text-based search engines have played a major role in research, communication, and publication in many academic fields. More recently, using advanced semantically-driven NLP technologies to solve various problems in science as well as the humanities is becoming more and more popular (Gábor et al., 2018). Relation extraction and classification allows for enhanced discoverabilty of scientific literature.
The results from traditional web search engines are too broad. The limited power of the standard search engine makes it difficult for researchers to acquire high quality information. Various approaches have been applied to extract fine-grained semantic information from scientific text to meet different needs. (Ronzano et al., 2016) initialized the Scientific Knowledge Miner Project that focuses on citation characterization and scientific article recommendation. Summarization of scientific papers has also been well studied. The information from citations plays an important role in the summarization task. (Mei and Zhai, 2008) made use of citation context to generate impact-based summaries. (Klampfl et al., 2016) aimed to summarize the relevant text from a reference paper based on another document that cites this reference paper. Abstracts also encode critical information about a paper. (Jin and Szolovits, 2018) presented a new neural approach to sentence classification in medical scientific abstracts utilizing context information. SemEval-2017 Task 10 (Augenstein et al., 2017) called for papers to work on the extraction of keyphrases and relations from scientific papers, which is closely related to our reproduction work. Recent study also shows the effectiveness to use neural approaches for relation extraction. Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) are commonly used neural architectures for the task (Lee et al., 2017; Li and Mao, 2019; Zhang et al., 2018).

3. Task and Data Description

The goal of SE18T7 was to extract and classify semantic relations from the abstracts of scientific publications from the CL community (Gábor et al., 2018). The challenges of this task are about semantic relations extraction from a specific domain. Overall, the relations instances in the data we will be working on were annotated with one of six relations, namely `USAGE`, `RESULT`, `MODEL`, `PART_WHOLE`, `TOPIC`, or `COMPARISON`. All of the relations except `COMPARISON` were asymmetric, so those instances could additionally be designated as X-`REVERSE`, forming 11 classes in total. The datasets for the task were downloaded from the official SE18T7 web page [6]. The data was provided with inline XML annotations.
Based on the subtasks (described later in this section), two training sets are provided separately, namely `D1.1` and `D1.2` where relations are annotated on clean and noisy entity annotations, respectively. Figure 1 shows the class frequency distribution of those two datasets. And here is an example of a training instance from `D1.1`.

```
<text id="H01-1058"> ... <abstract> ...
    The oracle knows the reference
   word string and selects the word
   string with the best performance (
   typically, word or semantic error
   rate) from a list of word strings
, where each <entity id="H01-1058.13">
   word string</entity> has been
   obtained by using a different <
   entity id="H01-1058.14">LM</entity>
    ... </abstract></text>
```

```
RESULT(H01-1058.13,H01-1058.14,REVERSE)
```

Listing 1: An example relation from the training data

Relation Type	Dataset		
	D1.1 ("clean" entities)	D1.2 ("noisy" entities)	**Total**
USAGE	296	323	619
MODEL-FEATURE	226	123	349
USAGE-R	187	147	334
PART_WHOLE	158	117	275
TOPIC	8	230	238
PART_WHOLE-R	76	79	155
MODEL-FEATURE-R	100	52	152
RESULT	52	85	137
COMPARE	95	41	136
RESULT-R	20	38	58
TOPIC-R	10	13	23
Total	1228	1248	**2476**

Table 1: Label distribution in the provided training data, sorted by total number of occurrences

Three subtasks are introduced to focus on different aspects of relation extraction. Subtask 1.1 (`T1.1`) is relation classification on *clean* data. Both in the training and testing data, entity mentions are manually annotated (`D1.1`). Then further in the training data, valid semantic relations between

[5] https://gitlab.com/brandeis-llc/reprolang2020-relation-extr.

[6] https://lipn.univ-paris13.fr/~gabor/semeval2018task7/

two entity mentions are also manually annotated. The goal of `T1.1`is to predict the correct semantic relations between the pre-annotated entity pairs in the test data.

Subtask 1.2 (`T1.2`) is relation classification on *noisy* data. The data and the goal of this subtask is similar to `T1.1`except that both in the training and testing data the entity mentions are automatically annotated (`D1.2`). The data is noisy in the sense that the boundaries of the entity mentions might be incorrect. In the training data, the valid semantic relations are still annotated manually with no regard to the potential boundary errors of entity mentions.

Subtask 2 (`T2`) is relation extraction and classification. The training data for this `T2`is identical to `T1.1`with both entity mentions and relations manually annotated (`D1.1`). However, the test data only contains entity mentions without relations between them. `T2`is in fact a two-phased process and the goals are to predict valid entity pairs first (`T2.E`) and then and the relation type of each pair (`T2.C`).

4. Original Work Summary

The ETH-DS3Lab system which we reproduce here opted a deep neural network method based on two widely-used neural network architectures – CNN and RNN (more specifically, Bidirectional Long Short-Term Memory (BiLSTM)) for all three subtasks in which their results ranked first. Concretely, they proposed an approach to combine the predictions from CNNs and RNNs as an ensemble model to perform relation classification and extraction simultaneously. They also chose the weighted cross-entropy as the objective function and upsampling strategy (for `T2`only) to alleviate the negative effect from the class unbalance.

In addition to the model architecture presented in paper, the authors describe various components for preprocessing and postprocessing steps taken in the development of the system that effectively raise the system performance. Specifically, in the preprocessing steps, they cropped the sentences to only keep the entity pairs and tokens in between, and cleaned the sentences by flattening nested entities and removing all tokens between parentheses and brackets (`(, ), [, ]`). Moreover they inserted entity tags to mark entity boundaries within a sentence, and added part-of-speech (POS) tags to provide extra syntactic information. Finally before feeding into the networks for training, all *long* sentences were dropped from the training data (because their experiments showed that long-distance relations introduced noise and only reduced the performance of the system), and synthesized training instances were added. Annotated labels were re-organized and transformed for training 6-way and 12-way supervised classifiers for two subtasks. For vector representation, ETH-DS3Lab system used pre-trained word2vec embedding as well as dense embeddings for POS and token orders trained on-the-fly. In the postprocessing steps, several rules such as no reversed relation of `COMPARE`, and each entity mention can only belong to one relation, are applied to ensure the validity of predictions.

To pre-train word2vec embedding, ETH-DS3Lab system also made use of external knowledge by collecting supplementary datasets for the training of word embeddings and a language model which is used to select more valid sentences as training samples.

Unfortunately, authors of RHZ didn't publish the codebase along with the paper. Thus we re-implemented the classification system following description from the paper as a blueprrint.

5. System Re-implementation

Figure 1 shows the overall architecture of our replicated relation extraction and classification system. The pipeline starts from downloading task data which represented as the grey box on top of the figure. Going from the left blue boxes, we use the provided training data as well as external corpora to train gensim word2vec (Řehůřek and Sojka, 2010) and Fasttext embeddings (Bojanowski et al., 2016) that can be seen as a part of the data vectorization process. Going from the right blue boxes is the beginning of the data preprocessing pipeline. It includes basic NLP such as sentence splitting and POS tagging. Then text manipulation and relation directionality strategy are applied to get clean version of data and reduce the output categories. Right before the vectorization, we augment the training data by upsampling and generating reliable synthetic data using language model. After the data is converted to vectors, we train multiple CNNs and RNNs to use ensemble method for getting the final classification results. In the rest of this sections we will discuss each replication step in detail and report various challenges.

5.1. Hardware and Software Specification

We used a single workstation equipped with 36-core Intel Xeon CPU (2.10GHz), 128GB RAM, and 4 × Geforce TITAN Xp (12 GB VRAM) GPUs, running Redhat Linux 7.4. The system was developed on python 3.6 and tensorflow 2.0. All software dependencies and external libraries are listed in the code repository.

5.2. Data Collection

In addition to the datasets provided as a part of the shared task (`D1.1`, `D1.2`), in the RHZ paper, the authors combined the ACL Anthology Reference Corpus (ACL-ARC) (Bird et al., 2008) with abstracts from papers posted on arXiv [7] `cs.CL` papers to train word embeddings (later used for vector representation of text data) and to train a language model (later used for validating synthetically generated data). When attempting to recreate the data set that the authors described, we encountered some challenges.

First off, there are multiple versions of the ACL-ARC and the RHZ does not specify which version was used for training or from which source the data was gathered. Instead, the paper reported an approximate number of tokens obtained from the ACL-ARC to be 90 million. The ACL-ARC homepage [8] lists three versions of the corpus with various formats available. Additionally, a cleaned version of the corpus is available for download from a third-part

[7] `https://arxiv.org/`

[8] `https://acl-arc.comp.nus.edu.sg/`, accessed at 2019-11-29

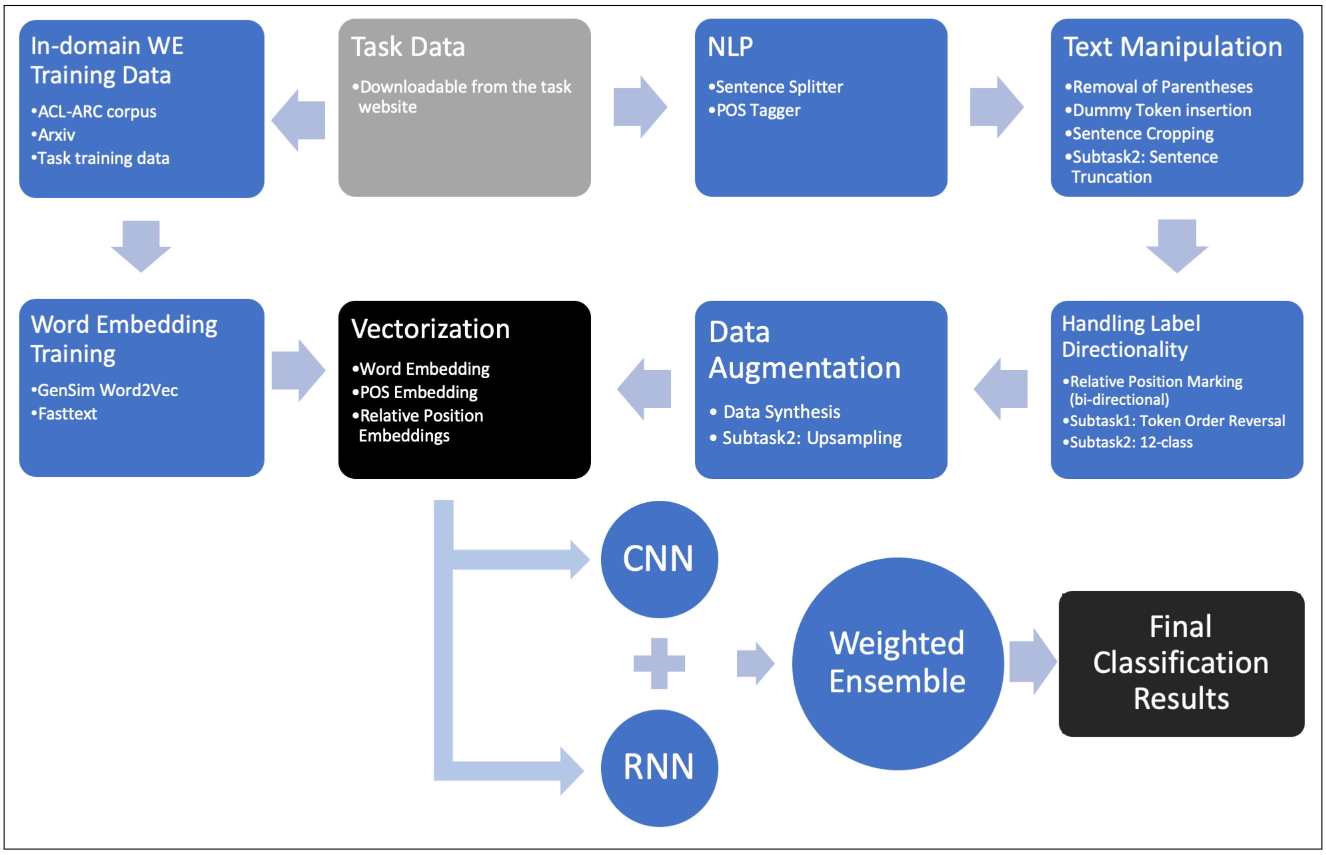

Figure 1: The overall architecture of the replicated relation extraction and classification system. Grey box on the top is the entry point of the pipeline. Pre-processing pipeline is finished at the vectorization component, and the resulting vectors are fed to neural classifiers to train or test.

website[9], that was used in Lahiri (2014). We downloaded multiple versions of the corpus and tokenized them using a simple whitespace tokenizer (using unix `wc` command). Due to potential differences in tokenization, we did not expect the token count to match exactly that which was reported in the RHZ, so we chose the version of the ACL-ARC that was closest to the token count reported in the paper. This was Version 20080325 downloaded from `https://acl-arc.comp.nus.edu.sg/` and had 87,176,836 tokens. . Also in the RHZ paper, the authors describe collecting all arXiv cs.CL abstracts since 2010, that resulted in approximately 1 million tokens. When we collected and tokenized abstracts fitting this description the resulting dataset contained approximately twice as many tokens as reported. We believe that is because of explosive increase in the number of CL publications in the open-access archive. We experimented with various cutoff dates in an attempt to replicate the data that would have been available at the time of publication of the ETH-DS3Lab system, and picked December 31, 2017 (the deadline of the SE18T7 was January 22). Our final arXiv data had 811,617 token.

5.3. Pre- and Post-processing Procedures

5.3.1. Basic NLP

Because a relation is confined in the shared task between two entities within a single sentence, it is very crucial step to split sentences from the provided XML annotation data before perform relation extraction (for `T2`). Failing to correctly segment sentences by over-segmenting results in some training instances being ill-formed due to the relation being split across separate instances. If the sentence splitter misses a split between sentences, spurious negative relation instances will be introduced. Unfortunately because the RHZ didn't reveal any details on how sentence segmentation is conducted, we experiment with different sentence segmentation pipelines as follows. In the first place, the relation-annotated dataset (`D1.1`, `D1.2`) was release in in-line XML format, so the first step was to convert the in-line annotation into character offset based standalone data structure for further cleaning and processing. Then, next step is to split sentences. We first experimented with the spaCy (Honnibal and Montani, 2017)[10] sentence segmenter, but found that many sentences were being erroneously split on periods and commas found in abbreviations such as "e.g." and "et al." These errors persisted across experiments with different spaCy pre-trained neural models[11] including `en_core_web_lg`, `en_trf_bertbaseuncased_lg`,

[9]`https://web.eecs.umich.edu/~lahiri/acl_arc.html`, accessed at 2019-10-24

[10]`https://spacy.io/`

[11]`https://spacy.io/models/en-starters`

and en_trf_xlnetbasecased_lg. Next we tried using the CoreNLP (Manning et al., 2014) [12] sentence splitter module, and found that it performed too conservatively for our data. Ultimately, we chose to use the CoreNLP sentence splitter module configured with a additional regex to split some sentences that were missed by the segmenter. There were at least three instances in which a sentence was incorrectly segmented due to the presence of punctuation within words. We suspect that these errors are present due to errors in original typing or errors from digitization old publications.

As POS tags used as features in the neural model, we also run the input data through the CoreNLP POS tagger to get text tagged, as reported in the RHZ paper.

5.3.2. Text Manipulation

Once the basic NLP modules processed the input data, next is to manipulate the text to fit more to the task-specific requirements. First described in the RHZ was removing text within parenthesis. The authors do not specify how they handle instances in which either entity is contained within parenthesis or situations where an entity contains an opening or closing parenthesis. We decided to remove parentheses that don't cross entity boundaries. Next, dummy tokens are used to indicate entity boundaries. A special "<e>" token is attached to before and after all entities in the annotation data. Additionally, number tokens (tagged as CD by the POS tagger) that were not part of proper nouns are replaced with a single wildcard token, "###".

Then each sentence is processed to generate training *instances* by flattening entities. As a result, an instance is a sentence that has only two entities annotated. For example, if the original sentence has three entities annotations (e1, e2, e3), three training instances are generated by choosing two out of three (<e1, e2>, <e1, e3>, <e2, e3>). Then, for T1, we only take entity pairs that explicitly annotated with labels in the training data. For T2, all entity pairs that are not annotated are taken as NONE-labeled negative training instances. The RHZ describe dealing with nested entities by flattening them, however it is unclear how the authors dealt with relations involving the inner entity of a nested entity. We simply consider all nested entities as independent entities, not differently from others.

Next is cropping and truncation. By cropping an instance, we remove all tokens before the first entity and after the second entity. Truncation is done only for relation extraction task (T2). As shown in the RHZ that long-distance relations are very seldom, all instances that have more than 19 tokens between two entities (excluding entity tokens) are treated as negative samples. There were a few places in the RHZ that were indicating 19 as "maximum sentences length", which can be interpreted as the count includes tokens from entities. However we decided to count excluding the entity tokens as described in the section 2.3 of the RHZ paper.

5.3.3. Label Transformation

The provided datasets are annotated with 11 classes (table 1). We followed the exact same steps described in the RHZ paper to transform class space to perform 6-way classification (for T1) and 12-way classification (for T2). Namely, for T1, we invert the order of tokens between two entities of REVERSE instances and treated the resulting token sequences as non-REVERSE instances. ([<e1> w1 w2 w3 <e2>, USAGE-R] $\rightarrow$ [<e1> w3 w2 w1 <e2>, USAGE], not inverting in-entity tokens). The RHZ states that having a full inverted text of those REVERSE relations can yield better performance, although the new generated "sentences" are not grammatical. This was also made possible only because the test instances were tagged as REVERSE if the relation goes from later entity to earlier entity, although they were not tagged with the actual relation labels. Eventually, by merging all REVERSE labels, T1became a 6-way classification task. For T2, because extracting related entity pairs was part of the task, test data didn't have such entity pair instances as in T1. Thus we could not apply the same token inversion. To solve extraction and classification simultaneously, we simply treated all entity pairs that are not annotated (and "long" instances) as having an additional NONE class. As a result, T2became a 12-way classification task. Using truncation (by *length*) previously described, NONE instances we obtained were $39,077$, different from $34,824$ as reported in the RHZ.

5.3.4. Data Augmentation

In training ETH-DS3Lab system, RHZ *exploited* provided data by 1) feeding D1.1+D1.2to both T1.1and T1.2classifiers, and 2) feeding D1.1+D1.2+"predictions of the system for T1.1and T1.2" to T2classifier. It is not so clear what exactly *predictions of the system* are from the description in the paper, however we interpreted it to be automatic labeling on test sets. Test sets for T1.1and T1.2both have 355 relation instances. We used our ensemble classifiers for T1.1and T1.2that showed the best performance to automatically label two test sets, and added the result (Dpred) to the training data for T2. Negative instances (non-relations) from T1test sets were not added to T2training set, as that would only worsen the class imbalance in training data.

We also synthesized additional training data (Dsyn) using the strategy described in the RHZ, and used them for both T1and T2tranining. That is, by combining entities from test data and tokens-between from training data. Gold-standard labels were also taken from training data. Then we filtered the synthetic sentences using a language model trained with the text data obtained in section 5.2. in training word embeddings, using the KenLM language modeling toolkit[13] (Heafield, 2011) (5-gram and the default smoothing options). We found that using the threshold in the RHZ paper (-21 log probability) resulted in far more synthetic instances (~33,000) than those reported by the RHZ (61). Thus, instead of using a threshold, we chose to rank the resulting instances based on language model probability and keep the top N instances to match the number of synthetic instances in the RHZ ($N = 61$).

For T2, because, after generating negative instances, the class distribution is largely skewed toward NONE, ETH-

[12] https://stanfordnlp.github.io/CoreNLP/, version 3.9.2

[13] https://github.com/kpu/kenlm, version 96d303c

DS3Lab system used upsampling to balance out the distribution. In the process of replicating, we implemented upsampling as simply duplicating positive instances keeping the class distribution ratio, but to match the sum of positive instances with the number of negative instances.
Label distributions during preprocessing are presented in table 2.

5.3.5. Post-processing

The ETH-DS3Lab system implements two post-processor. First one is only for `T1`, and the procedure is masking the only symmetrical class (COMPARE) for test instances already marked as `REVERSE`, which we successfully re-implemented. Second procedure is to discard additional predictions when two or more predictions put on a single entity, under the assumption that each entity could only be part of one relation. However, we could not find any justification for the assumption either in the RHZ or in the relation annotation process presented in Gábor et al. (2018). We also thought that, given imbalance in the class distribution, taking only one positive prediction and treating the rest as negative instances can affect the evaluation metrics (concretely, macro-F1 score) in an unexpected way. In the end, we decided not to implement such a component.

5.4. Neural Network Architecture

The main architecture of the system presented by RHZ is an ensemble of CNNs and Bidirectional RNNs. The input is a concatenation of dense representations of word tokens, POS and positional indices (forward and backward). The output probability distribution vectors after softmax layer from each neural model will be averaged using a weighting schema that puts more weights on RNN predictions for longer sentences before the final predictions are made.
Embeddings Our embedding settings strictly follows the RHZ paper. We trained word2Vec embeddings using the training data for the original task as well as external data source from arXiv and ACL-ARC. The total number of tokens as reported in RHZ for training embeddings is 91,304,581 . However, it is difficult for us to replicate the exact number count due the challenges we mentioned in section 5.2.. The size of our text corpus for word embedding training is 88,070,855 tokens.
In addition to using pre-trained word embeddings, we also created dense representations for POS tags and relative positions of each token in a sequence. The initial values for all the embeddings except pre-trained word2vec are randomly initialized from a uniform distribution ranging from -1.0 to 1.0. In practice, word tokens that cannot be looked up in the pre-trained embedding table will also be randomly initialized.
CNN Classifier The RHZ stated that the structure of their CNN model follows Kim (2014) and Collobert et al. (2011), hence we decided to reproduce the multichannel CNN model by Kim (2014) that was originally for sentence classification, which improved on the state-of-the-art of several text classification tasks. It composed of multiple embedding layers, which is followed by multiple concurrent convolutional layers with different kernel sizes, and max-over-time pooling layers. The intermediate output from pooling layers is concatenated and flattened before fed into a fully connected layer with dropout and softmax layer. Kim (2014) also experimented with different variants of the CNN model by using static embeddings, non-static embeddings or the combinations of both. Which setting yields better performance is not conclusive and depends on characteristics of data. However, this detail was not mentioned by RHZ. In the end, we decided to use non-static embeddings only given that, except word embeddings, POS embedding and relative position embeddings are all randomly initialized at the first place. We needed to set them trainable and made the embeddings more domain-specific to our task.
RNN Classifier The RNN model presented in RHZ is also composed of multiple embedding layers that are followed by a forward LSTM layer and a backward LSTM layer. Both LSTM layers are dynamic and can mask reserved padding tokens, which works well on sequences with variant lengths. The intermediate output from the final hidden states of LSTM layers is concatenated and fed into a fully connected dense layer with dropout and softmax layer.
Ensemble of CNN and RNN We also followed the ensemble system of CNN and RNN classifiers proposed by RHZ. Essentially, each classifier will be trained multiple times. Then we compute the average of the probability distribution from each output to reduce the variance of the results from a single run. On top of this, we also combined the predictions from CNN and RNN by assigning different weight to them using the $\sin$-based weighting formula proposed by RHZ. The intuition is based on the observation that RNN classifier tends to outperform CNN classifier on longer sequences, while CNN classifier has the ability to capture local context information within a shorter sequence. Thus, for longer sequences we will assign higher weight to the predictions from RNN. Similarly we also adopted the weighted cross-entropy as the loss function for our models since the training data of each category is highly unbalanced. Specifically in our reproduction work, at each run we set the random state by selecting the number of the random seed from a range of 0 to 10000 to ensure the train and validation data is split without being introduced too much human bias.

5.4.1. Model Configurations and Hyperparameters

As a reproduction work, our goal was to follow the original settings and see if that produces the same results. However, we were facing a fair number of challenges from the change of data source, lack of details and the non-deterministic nature of deep learning approaches.
In our experiments, we followed values from the final configuration in the Table 2 of the RHZ paper. Many of the parameters we adopted are the same with them, including the size of different embeddings and parameters of neural model architecture. We additionally made the following assumptions and changes to our best knowledge. Table 3 shows the difference between our parameter values and theirs.

- **Initial learning rate** Original learning rate was 0.01 across all classifiers. In our experiments, the value was too high for the model to converge (even with Adam optimization), and it caused F1 score to be ill-defined

Relation Type	D1.1+D1.2	RHZ	+Dsyn	+invert	**+invert +Dsyn**	+truncate	+Dsyn +truncate	+Dsyn +Dpred +truncate	**+Dsyn +Dpred +truncate +upsample(1.0)**
USAGE	619	619	634	953	**979**	614	629	826	**10005**
MODEL-FEATURE	349	349	365	501	**518**	349	365	466	**5611**
USAGE-R	334	334	345			330	341	457	**5535**
PART_WHOLE	275	275	282	430	**438**	275	275	361	**4372**
TOPIC	238	238	238	261	**261**	238	238	304	**3682**
PART_WHOLE-R	155	155	156			155	155	210	**2543**
MODEL-FEATURE-R	152	152	153			152	152	177	**2144**
RESULT	137	137	141	195	**199**	133	133	173	**2095**
COMPARE	136	136	142	136	**142**	133	133	165	**1998**
RESULT-R	58	58	58			58	58	61	**738**
TOPIC-R	23	23	23			23	23	26	**314**
NONE	47998	34824				39077	39077	39077	**39077**
Total	2476	37300	2537	2476	**2537**	41537	41579	42303	**78147**

Table 2: Label distribution after each preprocessing step. Bold-faced columns are data used for training for T1and T2.

Parameter	**Final value**	**Original Value**
Word embedding dimension	200	200
POS embedding dimension	30	30
Positional embeddings dimension	20	20
Number of CNN filters	192	192
Size of CNN filters	2-7	2-7
Regularization parameter (λ)	0.01	0.01
Number of LSTM units (RNN)	600	600
Dropout ratio (CNN and RNN)	0.5	0.5
Ensemble size	20	20
Training batch size	64	64
CNN Initial learning rate	**0.001**	0.01
RNN Initial learning rate	**0.005**	0.01
Number of epochs (T1.1)	**~9.9**	200
Number of epochs (T1.2)	**~7.6**	200
Number of epochs (T2)	**~2.7**	10
Upsampling ratio (T2)	1.0	1.0

Table 3: The comparison of our final parameter values and the values from original paper. Values in bold means ours are different from original values.

during the training. With our implementation of grid search (described below), we found that optimal learning rates are much lower to 0.005 for RNN classifiers and 0.001 for CNN classifiers.

- **Number of epochs** Original number of epochs are fixed (200 for T1and 20 for T2). It wasn't clear enough for us that the number of epochs was for a single run or for the ensemble classifiers. Instead, as we used mini-batch gradient descent, we counted *batches* rather than epochs during the training. At each batch the model will consume fixed number (64) of training instances from the data pipeline that *infinitely* pulls batches by repeating input dataset until the model converges. We also adopted the early stopping approach, meaning that our model will stop training if the macro F1 score on the validation set did not improve over 100 steps reducing the training time. In this way it is guaranteed to train the model well and leave less carbon footprint. The numbers presented in table 3 are conversion from average number of batches consumed by each network in the ensemble multiplied by the batch size and divided by the size of the input dataset.

- **Padded sequence length** Before the data is ready to be passed into the model, it needs to padded to the same length. It isn't difficult to implement but non-trivial nevertheless. If the padded length is too long, shorter sequences will become very sparse. While if it is too short, some essential information will be cut off for longer sequence. Although RHZ paper didn't provide any details of padding size, to replicate the experiments as closely as possible, we stick to the strategy to pad all the sequences to the length of the longest sequence in the training data, which is 58 for T1and 35 for T2in the end.

- **Grid search for hyperparameters** RHZ also stated about performing grid search but presented parameter search space without much detail. We estimated performing full grid search with given parameter space requires 36,720,000,000 iterations of training cycles, in each of which the system requires multiple networks trained for ensemble voting. Although training a single network given the size of data (2537 instances for T1, ~78k instances for T2) didn't take very long time in our experiments (11.7 seconds for CNN and 24.1 seconds for RNN with T1setting under single GPU configuration, and 103.2 and 311.5 seconds with T2setting), 36 trillion iterations do not sound practical. We implemented grid search, but experiments only with varying · $L2$ regularization λ [0.01, 0.1], · CNN and RNN learning rates [0.001, 0.005, 0.01, 0.05, 0.1], · word embedding dimensions [100, 200, 300], and and found optimal hyperparameters listed in table 3

6. Results

In this section, we presents our experiments results. Table 4 reports the overall results for each Subtask along with the original F1-score which corresponds to the Table 3 in

Subtask (word2vec)	P	R	F1
1.1	80.94	81.55	80.67 (81.7)
1.2	83.07	86.37	83.81 (90.4)
2.E	30.21	58.04	39.74 (48.8)
2.C	35.11	64.11	43.13 (49.3)

Table 4: The overall Precision (P), recall (R) and F1-score (F1) on the test set (in %) for each Subtask. This is the reproduced results of the Table 3 in the original paper. The F1-score from the original paper is reported in the parenthesis for easy comparison

the original paper. For `T1.1`, we achieved F1-score 80.67 compared with 81.7 reported originally. The experiment setting for `T1`is relatively simple and clear among all four tasks. Thus the reproduction work can be proceeded without being introduced much bias by us, and the results are the closest to original ones.
For `T1.2`and `T2`, our F1-scores are lower than the original ones by about 7%. This discrepancy might be resulted from multiple factors. Apart from the difference in text preprocessing and model hyperparameter tweaking, we believe the most different part is from the use of external data resources and synthetic data samples. First the corpora we collected for training embeddings might be of different version, since the there are more CL publications being added into the corpora. We cannot be sure about the exact publication date before when we should stop the data collection, so instead we have to resort to our best guess. The upsampling schema proposed in the original paper also didn't work as well as expected. We think the performance boost from repeated data is limited. The original paper authors also mentioned that they added the predictions for Subtasks 1.1 and 1.2 as external training data for `T2`. However, since our performance for `T1`is lower than the original ones, the additional predication errors might also propagate.
Table 5 reports the detail scores for each relation type for `T1.1`. This is corresponded to the Table 4 in the original paper. Although our overall result for `T1.1`is similar to theirs, the distributions over some relation types is different. In the orginal paper, the recall for `TOPIC` is only 0.50 while

Relation type	P	R	F1
COMPARE	80.95	80.95	80.95 (97.56)
MODEL-FEATURE	71.27	71.21	71.21 (72.59)
PART-WHOLE	78.57	78.57	78.57 (79.43)
RESULT	93.33	70.00	80.00 (77.78)
TOPIC	75.00	1.000	85.71 (66.67)
USAGE	86.59	88.57	87.57 (87.36)
Micro-averaged total	81.97	81.97	81.97 (82.82)
Macro-averaged total	80.94	81.55	80.67 (81.72)

Table 5: The Precision (P), recall (R) and F1-score (F1) on the test set (in %) for each relation type for Subtask 1.1. This is the reproduced results of the Table 4 in the original paper. The F1-score from the original paper is reported in the parenthesis for easy comparison

we are having 0.75. They also reached the 1.0 precison for `COMPARE`, but ours is around 0.8.
Considering various of randomness factored in the neural pipeline, we think the difference of scores here is reasonable.
Due to the time limit, we are more focused on mimicking the original experiments as closely as possible. During the study, we found training word2vec word embedding from fairly large corpus ($\sim$88 million tokens) is quite time-consuming (48 - 51 minutes without using GPU). Alternatively, we experimented with training word embeddings using FastText, which is much faster (12-18 minutes under identical configuration) and able to compute word representations for rare words or words that did not appear in the training data (Bojanowski et al., 2016). This might be helpful in scientific fields where many technical terms exist. In table 6, we reported the results for each Subtask. We followed the same settings except using FastText embeddings instead of word2vec. The macro F1-score is 0.79, which is slightly lower than using word2vec. We trained our FastText with the default setting. In the future, we will try to fine-tune it and see if we can make use of other embeddings.

7. Lessons Learned

The various challenges we encountered made it difficult to determine the source of problems in our system that caused it to fail to exactly replicate the RHZ results. We were grateful that the original authors reported token counts for their data collection procedure as this allowed us to be more confident that our collected data resembled the original data. It may be impossible to reproduce identically the collected data. Abstracts on arXiv that were available at the time the original authors collected their supplementary data, could have been deleted or modified in the years since. The system described in the original paper relies on multiple NLP tools for sentence segmentation, POS tagging, and language modeling. In section 4., we discuss challenges related to sentence segmentation. For POS tagging, we used Stanford CoreNLP version 3.9.2, which included among other updates, new POS models for English. This version was released 2018-10-05, which is after the publication of the original paper. We can safely assume that the original authors were using a prior version of Stanford CoreNLP, but cannot specify which version was used.
Our reproduction effort would have been made easier by more specificity with respect to the data collection procedures, or alternatively, if the original supplementary dataset were available for download. In addition, preprocessing and cleaning the data would be facilitated with instructions or examples for how to handle edge cases.

8. Conclusion

In this paper, we reported our replication study on semantic relation extraction and classification task on science papers originally published in Rotsztejn et al. (2018). By re-implementing the data pipeline and classifier system, and by comparing experiment results under similarly (if not exactly) configured experiments, we showed some important implementational details were left unknown or ambiguous

Subtask (FastText)	P	R	F1
1.1	79.16	81.34	79.16
1.2	75.93	78.52	76.58
2.E	27.19	68.39	38.91
2.C	28.03	57.25	33.75

Table 6: The overall Precision (P), recall (R) and F1-score (F1) on the test set (in %) for each subtask using pretrained FastText embeddings

in the RHZ paper. Although we have achieved fairly close results for some subtasks, others showed significant disparity that cannot be explained only with statistical perturbation in the algorithms. Throughout the discussion in the paper, we also showed specifics of challenges we faced partly due to those under-specified technicality, and decisions we made on our own with every detail. We believe that this and other replication study help promoting better practice in reporting scientific experiments in the CL and NLP community.

Acknowledgment

This research was funded in part by DTRA grant HDTRA1-16-1-0002, and a contract with the US Defense Advanced Research Projects Agency (DARPA), Contract CwC-W911NF-15-C-0238.

References

Anderson, J. E., Aarts, A. A., Anderson, C. J., Attridge, P. R., Attwood, A., Axt, J., Babel, M., Bahník, Š., Baranski, E., Barnett-Cowan, M., et al. (2015). Estimating the reproducibility of psychological science. *Science*, 349(6251).

Augenstein, I., Das, M., Riedel, S., Vikraman, L., and McCallum, A. (2017). SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada, August. Association for Computational Linguistics.

Begley, C. G. and Ellis, L. M. (2012). Raise standards for preclinical cancer research. *Nature*, 483(7391):531–533.

Bird, S., Dale, R., Dorr, B., Gibson, B., Joseph, M., Kan, M.-Y., Lee, D., Powley, B., Radev, D., and Tan, Y. F. (2008). The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA).

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Branco, A., Calzolari, N., and Choukri, K. (2016). 4REAL Workshop: Workshop on Research Results Reproducibility and Resources Citation in Science and Technology of Language.

Branco, A., Cohen, K. B., Vossen, P., Ide, N., and Calzolari, N. (2017). Replicability and reproducibility of research results for human language technology: Introducing an LRE special section.

Branco, A., Calzolari, N., and Choukri, K. (2018). 4REAL 2018 Workshop on Replicability and Reproducibility of Research Results in Science and Technology of Language.

Collaboration, O. S. et al. (2015). Estimating the reproducibility of psychological science. *Science*, 349(6251):aac4716.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Fokkens, A., van Erp, M., Postma, M., Pedersen, T., Vossen, P., and Freire, N. (2013). Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1691–1701, Sofia, Bulgaria, August. Association for Computational Linguistics.

Gábor, K., Buscaldi, D., Schumann, A.-K., QasemiZadeh, B., Zargayouna, H., and Charnois, T. (2018). SemEval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688, New Orleans, Louisiana, June. Association for Computational Linguistics.

Hagen, M., Potthast, M., Büchner, M., and Stein, B. (2015). Webis: An ensemble for twitter sentiment detection. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 582–589.

Heafield, K. (2011). KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July. Association for Computational Linguistics.

Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Ivie, P. and Thain, D. (2018). Reproducibility in scientific computing. *ACM Computing Surveys*, 51:63:1–63:36.

Jin, D. and Szolovits, P. (2018). Hierarchical neural networks for sequential sentence classification in medical scientific abstracts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3100–3109, Brussels, Belgium, October-November. Association for Computational Linguistics.

Johnson, R., Watkinson, A., and Mabe, M. (2018). *The STM report*. International Association of Scientific, Technical and Medical Publishers.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Klampfl, S., Rexha, A., and Kern, R. (2016). Identifying referenced text in scientific publications by summarisation and classification techniques. In *BIRNDL@JCDL*.

Lahiri, S. (2014). ACL ARC Style Browser. http://ec2-54-186-204-149.us-west-2.compute.amazonaws.com/acl_arc_style_browser/.

Lee, J. Y., Dernoncourt, F., and Szolovits, P. (2017). MIT at SemEval-2017 task 10: Relation extraction with convolutional neural networks. *arXiv preprint arXiv:1704.01523*.

Li, P. and Mao, K. (2019). Knowledge-oriented convolutional neural network for causal relation extraction from natural language texts. *Expert Systems with Applications*, 115:512–523.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mei, Q. and Zhai, C. (2008). Generating impact-based summaries for scientific literature. In *Proceedings of ACL-08: HLT*, pages 816–824.

Mieskes, M., Fort, K., Névéol, A., Grouin, C., and Cohen, K. (2019). Community perspective on replicability in natural language processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 768–775, Varna, Bulgaria, September. INCOMA Ltd.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. http://is.muni.cz/publication/884893/en.

Ronzano, F., Freire, A., Saez-Trumper, D., and Saggion, H. (2016). Making sense of massive amounts of scientific publications: the scientific knowledge miner project. In *Proceedings of the Joint Workshop on Bibliometric-enhanced Information Retrieval and Natural Language Processing for Digital Libraries (BIRNDL)*, pages 36–41, June.

Rotsztejn, J., Hollenstein, N., and Zhang, C. (2018). ETH-DS3Lab at SemEval-2018 task 7: Effectively combining recurrent and convolutional neural networks for relation classification and extraction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 689–696, New Orleans, Louisiana, June. Association for Computational Linguistics.

Zhang, Y., Lin, H., Yang, Z., Wang, J., Zhang, S., Sun, Y., and Yang, L. (2018). A hybrid model based on neural networks for biomedical relation extraction. *Journal of biomedical informatics*, 81:83–92.