SWE 522 Software Requirements Engineering

Research Report

Group G - Salih Yavuz - Uğur Sevcan - Deniz Baran Aslan

Introduction

In the research paper we have picked, "*Automatic Detection of Causality in Requirement Artifacts: The CiRA Approach*", Fishbach et al. present a new methodology for detecting causality relations in natural language data. They use the working definition of causality as a relation of one event triggering another event. The detection consists of two stages: identifying whether or not there is any causation in the first place, and then identifying the relation of the events.

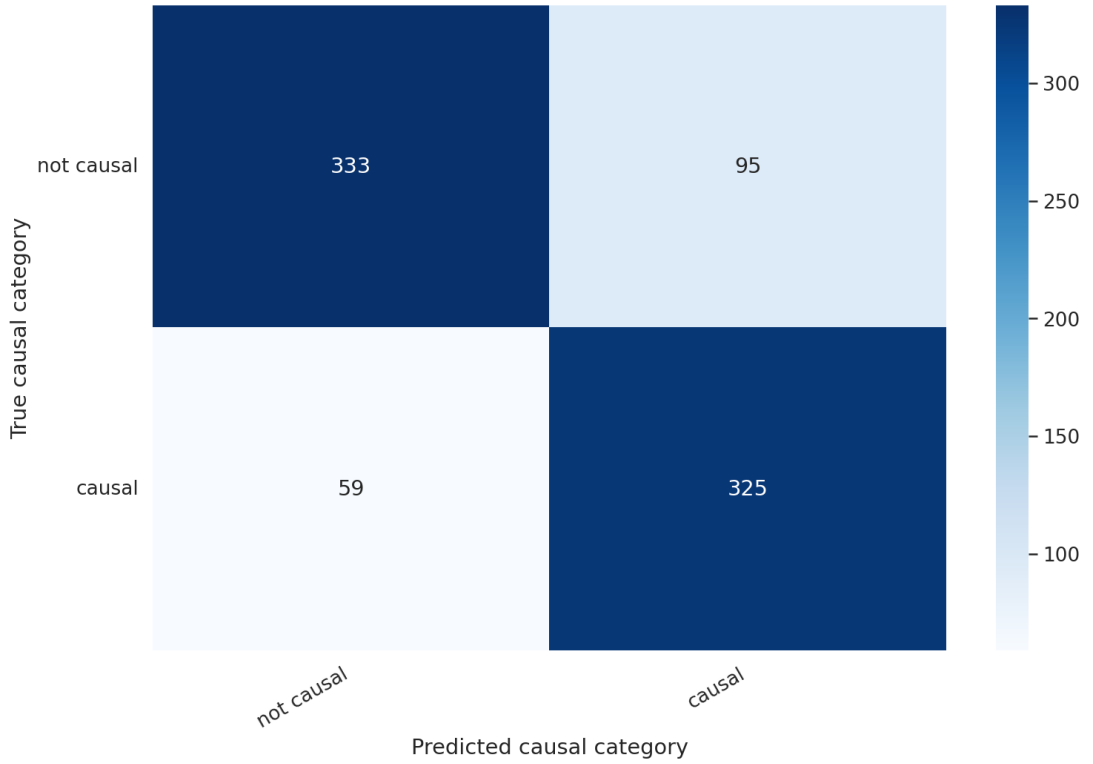
Initially, the researchers conduct a case study using almost 15,000 sentences from various requirement documents, each annotated by a team. This study allows them to identify common patterns, categorize causation relations, and provides them with more insight for the next step, automated detection. Here, the researchers employ three types of approaches: In the first method (rule-based approach), they simply iterate through the sentences and check if they include a regular expression such as "if, as, because", etc. In the second approach (machine learning), they employ several supervised ML models, using algorithms such as Naive Bayes, Decision Trees, and K-Nearest Neighbor. In the third approach (deep learning), they employ BERT. They tune BERT in three configurations: $BERT_{Base}$, $BERT_{POS}$, and $BERT_{DEP}$. The following table, taken directly from the paper, illustrates to which degree each method successfully detected causality:

			Causal (Support: 435)			**Not Causal (Support: 408)**			
		Best hyperparameters	Recall	Precision	F1	Recall	Precision	F1	Accuracy
Rule based		-	0.65	0.66	0.66	0.65	0.63	0.64	0.65
ML based	NB	alpha: 1, fit_prior: True, embed: BoW	0.71	0.7	0.71	0.68	0.69	0.69	0.7
	SVM	C: 50, gamma: 0.001, kernel: rbf, embed: BoW	0.68	0.8	0.73	0.82	0.71	0.76	0.75
	RF	criterion: entropy, max_features: auto, n_estimators: 500, embed: BoW	0.72	**0.82**	0.77	**0.84**	0.74	0.79	0.78
	DT	criterion: gini, max_features: auto, splitter: random, embed: TF-IDF	0.65	0.68	0.66	0.67	0.65	0.66	0.66
	LR	C: 1, solver: liblinear, embed: TF-IDF	0.71	0.78	0.74	0.79	0.72	0.75	0.75
	AB	algorithm: SAMME.R, n_estimators: 200, embed: BoW	0.67	0.78	0.72	0.8	0.7	0.75	0.74
	KNN	algorithm: ball_tree, n_neighbors: 20, weights: distance, embed: TF-IDF	0.61	0.68	0.64	0.7	0.63	0.66	0.65
DL based	$BERT_{Base}$	batch_size: 16, learning_rate: 2e-05, weight_decay: 0.01, optimizer: AdamW	0.83	0.80	0.82	0.78	0.82	0.80	0.81
	$BERT_{POS}$		0.82	0.76	0.79	0.71	0.83	0.77	0.78
	$BERT_{DEP}$ (CiRA)		**0.85**	0.81	**0.83**	0.79	**0.84**	**0.81**	**0.82**

Methodology

The data and the algorithms were made available online in a GitHub repository. In order to replicate the results, we had to make certain alterations to the code and fix bugs. Initially, the import and install statements for the Python modules were not compatible with our local environment so we adjusted them accordingly. We also modified some classes in order to satisfy the type requirements for the methods used in training. We chose to replicate the results the scholars achieved with their third approach, using BERT_{DEP}, as this was the final and definitive algorithm provided, and the most successful one overall. The outcome is illustrated below, as a table and a confusion matrix:

	precision	recall	f1-score	support
not causal	0.85	0.78	0.81	428
causal	0.77	0.85	0.81	384
accuracy			0.81	812
macro avg	0.81	0.81	0.81	812
weighted avg	0.81	0.81	0.81	812



For comparison's sake, we also replicated the results obtained through the first approach, the rule-based algorithm. The figures are provided below:

	precision	recall	f1-score	support
not causal	0.67	0.68	0.67	428
causal	0.64	0.62	0.63	384
accuracy			0.65	812
macro avg	0.65	0.65	0.65	812
weighted avg	0.65	0.65	0.65	812

These findings are in line with those reported in the paper, where BERT produces much more accurate results than simple regex detection. Note that an exact match is not possible since the researchers make use of Random Undersampling. This is done to offset the imbalance in the data, where non-causal sentences outnumber the causal. As such, even though we draw from the same dataset, we are not using the exact same causal entries.

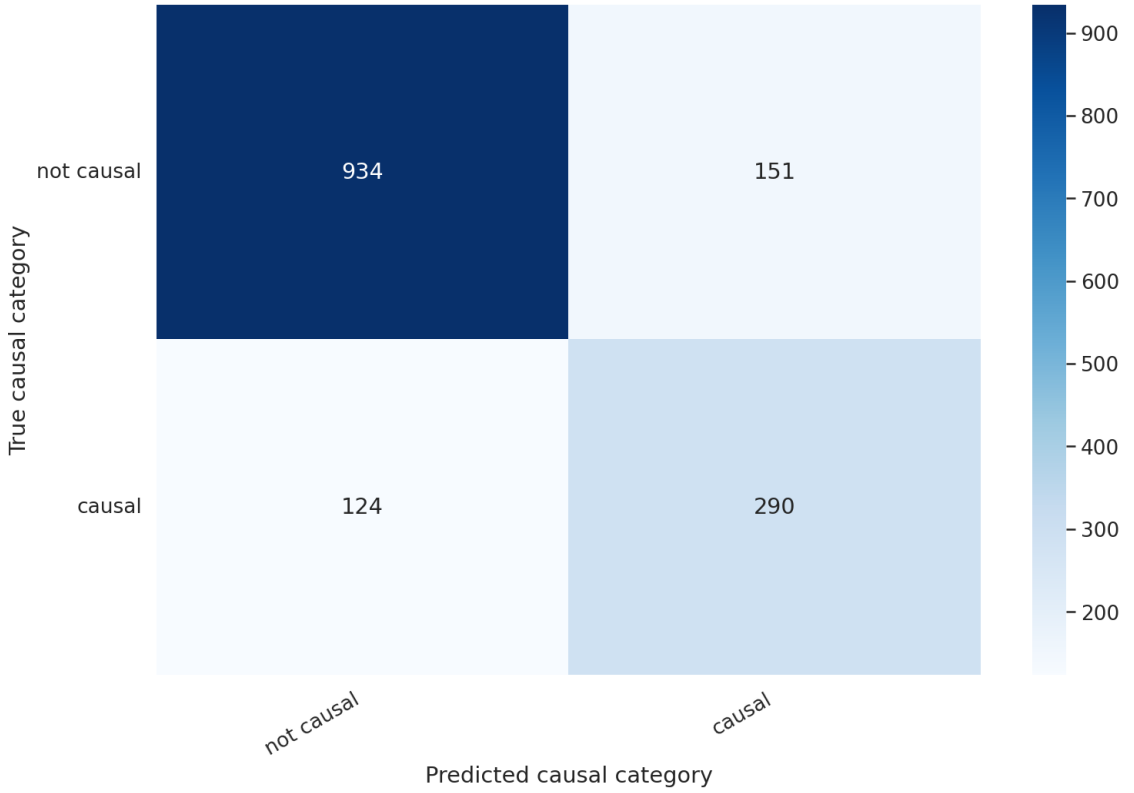
After our analysis, we came up with two ideas to improve upon the results. First, we used random sampling instead of undersampling, making the selection truly random. Even though an imbalanced dataset is not preferable, we hypothesized that feeding a larger amount of the available data to the algorithm would produce more accurate results. This new iteration went from processing **812** sentences to **1499**. Our second improvement was aimed at time efficiency rather than accuracy. The initial replication algorithm took **2 hours 33 minutes 12 seconds** to execute, only for the $BERT_{DEP}$ configuration. Adding the time cost of bug-fixing and implementation, this made it infeasible to try and replicate the other configurations available. As such, we decided to lower the number of epochs used in model training from 20 to 8. The reason we chose to stop at 8 is that anything lower resulted in less than 99% accuracy, while anything higher only led to diminishing returns. This can be observed in the console output below, produced after the initial replication:

Epoch 1/20	Train loss 0.5450814715380152	accuracy 0.7229064039408867
Epoch 2/20	Train loss 0.36568901794297354	accuracy 0.853448275862069
Epoch 3/20	Train loss 0.2374447045430284	accuracy 0.9177955665024631
Epoch 4/20	Train loss 0.16042009303392155	accuracy 0.9550492610837439
Epoch 5/20	Train loss 0.11796901462940279	accuracy 0.9699815270935961
Epoch 6/20	Train loss 0.07999318897474117	accuracy 0.9818349753694582
Epoch 7/20	Train loss 0.05596016564504232	accuracy 0.9866071428571429
Epoch 8/20	**Train loss 0.03520396247051233**	**accuracy 0.9910714285714286**
Epoch 9/20	Train loss 0.02883554566859109	accuracy 0.9936884236453202
Epoch 10/20	Train loss 0.02405306263312811	accuracy 0.9947660098522167

Epoch 11/20	Train loss 0.01539155695935209	accuracy 0.9964593596059114
Epoch 12/20	Train loss 0.021956920634442337	accuracy 0.9958435960591133
Epoch 13/20	Train loss 0.008000959884282503	accuracy 0.997844827586207
Epoch 14/20	Train loss 0.006473736008306982	accuracy 0.9979987684729064
Epoch 15/20	Train loss 0.004606794334554946	accuracy 0.9987684729064039
Epoch 16/20	Train loss 0.006222344957234159	accuracy 0.9987684729064039
Epoch 17/20	Train loss 0.001817557637922856	accuracy 0.9992302955665024
Epoch 18/20	Train loss 0.0004075719102057595	accuracy 0.9998460591133005
Epoch 19/20	Train loss 0.0007637592518844694	accuracy 0.9995381773399015
Epoch 20/20	Train loss 0.0020092691606379974	accuracy 0.9995381773399015

Our new implementation managed to train the model in only **28 minutes 26 seconds.** It is worth noting that since the algorithm was run on a Google Colab instance, the availability of system resources at different times might have had an effect on this change as well, specifically the allocated GPU. The results of the new implementation are as follows:

	precision	recall	f1-score	support
not causal	0.88	0.86	0.87	1085
causal	0.66	0.70	0.68	414
accuracy			0.82	1499
macro avg	0.77	0.78	0.78	1499
weighted avg	0.82	0.82	0.82	1499



Discussion and Conclusion

Comparing the results of our replication with those of our new implementation shows that the algorithm is now better at detecting non-causal sentences by every measure, but worse at detecting causals. The outcome indicates that feeding more data of a certain type to the algorithm leads to better detection of that type, and the inverse also applies. Since the dataset contained a greater amount of non-causal entries, the random sampling picked more non-causals as well. This outcome was within expectations, and partially affirms our initial hypothesis.

Using the information and the insight provided by Fishbach et al., we were able to implement automatic causality detection in a more time-efficient way, and observe how unsymmetric input leads to unsymmetric output as well.

Our implementation, as well as other relevant files, are available online on GitHub.

References

Fischbach, J. et al. (2021). Automatic Detection of Causality in Requirement Artifacts: The CiRA Approach. In: Dalpiaz, F., Spoletini, P. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2021. Lecture Notes in Computer Science(), vol 12685. Springer, Cham. https://doi.org/10.1007/978-3-030-73128-1_2