Neural Machine Translation for Fact-checking Temporal Claims

Marco Mori[1], Paolo Papotti[2], Luigi Bellomarini[1], and Oliver Giudice[1]

[1]Bank of Italy, Italy
[2]EURECOM, France

Abstract

Computational fact-checking aims at supporting the verification process of textual claims by exploiting trustworthy sources. However, there are large classes of complex claims that cannot be automatically verified, for instance those related to temporal reasoning. To this aim, in this work, we focus on the verification of economic claims against time series sources. Starting from given textual claims in natural language, we propose a neural machine translation approach to produce respective queries expressed in a recently proposed temporal fragment of the Datalog language. The adopted deep neural approach shows promising preliminary results for the translation of 10 categories of claims extracted from real use cases.

1 Introduction

False and misleading information spreading on media negatively impacts our society by affecting people opinions and behaviours. To oppose such phenomena, *fact-checking* is the process aiming at verifying the correctness of facts in a piece of text, i.e., *claims* (Nakov et al., 2021). To this end, claims have to be debunked against trustworthy structured and non-structured sources. In this work, we report on our effort on statistical claims in collaboration with a national central bank in Europe. Specifically, we focus on claims about economic events and trends that should be checked against temporal data sequences, namely *time series*, the standard format for time-based data in this domain.

Fact-checking organizations debunk such claims applying a typically manual process to extract claims from text, retrieving pieces of evidence from relevant sources, and finally, reaching a verdict using such evidence. Given the large amount of information produced and shared online, this process cannot guarantee that an adequate number of economic claims is debunked. To enable scalability, statistical claims could be automatically verified

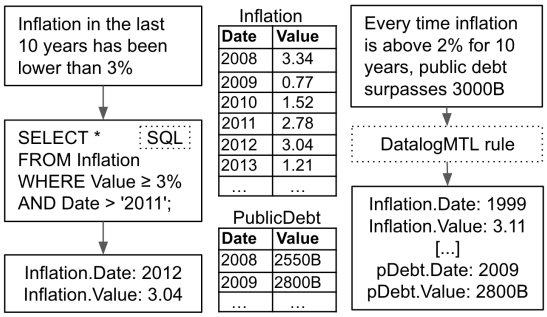

Figure 1: Examples of a claim verified with a SQL query (left) and a complex claim tested with a Datalog rule (right). In both cases, the result is not empty, thus the claim is false based on the evidence in the time series.

to support checkers in their work. In order to be interpretable to the human, not only should the automatic verification provide a label for the given claim, but also a clear explanation of the adopted reasoning process as well as the evidence used to draw the conclusion.

Declarative languages offer a valid solution to both challenges. Mature data modeling languages balance computational complexity and expressive power, addressing the scalability challenges. Interpretability is a major advantage of declarative languages, with well-known semantics applied to the ground data in a deductive, top-down approach (Hansen and Rieger, 2019).

Different solutions translate text formats to executable high-level languages, e.g., SQL queries, typically through neural architectures (Katsogiannis-Meimarakis and Koutrika, 2021; Saeed and Papotti, 2021). While these approaches exhibit promising results, they do not effectively translate claims including temporal aspects into an executable form. This depends on their limited capability in capturing the temporal logic of claims.

Example. Consider a central bank that offers a service to fact-check economic claims. Reference information is available in the standard format of time-series. The bank has data about economic

Proceedings of the Fifth Fact Extraction and VERification Workshop (FEVER), pages 78 - 82

measures, such as inflation, public debt, employment rate, interest rates, and about events such as economic crises, and salary increments. The example in Figure 1 shows the inflation and public debt metrics with key-value tables.

While some claims can be easily written as SQL queries for their verification, for most of them this is very difficult, or unfeasible in some cases. Consider the claims *(i)* "Inflation in the last 10 years has been lower than 3%" and *(ii)* "Every time inflation is above 2% for 10 years, public debt surpasses 3000B". A possible SQL query to verify claim *(i)* through counter-example detection is reported in the left-hand side of Figure 1. For Claim *(ii)*, a script should evaluate, for each year in the dataset, the value of public debt and then check a condition over inflation for the previous 10 years. Although such expression can be written in SQL for one given year, our claim requires checking the condition for every possible year, which turns out to be unfeasible or extremely laborious and inefficient in SQL. With DatalogMTL (Brandt et al., 2018), a Datalog fragment that incorporates temporal operators, we can represent our expression as:

$$\bot := \boxminus_{[0,10]}\, \mathit{inflationGt}(y), y = 0.02,\\ \mathit{public\_debt}(x), x \leq 3000B$$

The rule intercepts the cases in which Claim (*ii*) is violated. In fact, it is triggered whenever the body (right-hand side) is satisfied, meaning that the inflation is found greater than 2% for the 10 preceding years but the public debt is below or equal to 3000B. Specifically, the logical premise of our claim ("Every time inflation is above 2%") is checked through the temporal operator $\boxminus_\varrho$, i.e., "*always true in the interval* $\varrho = [t - 10y, t]$", so that for the rule to be triggered at a point in time t, the logical conclusion of our claim must be false, i.e., $\mathit{public\_debt}(x), x \leq 3000B$. The evaluation of the rule checks the claim and produces human-understandable counter-examples (Figure 1).

In this paper, we propose a neural machine translation approach to fact-check numerical temporal claims. In particular, our approach synthesizes temporal data queries (rules) from input claims for the verification of temporal claims. While there have been efforts to use neural architectures for generating SQL scripts (Yin et al., 2021), to the best of our knowledge, there is no support to translate textual claims into Datalog rules (Brandt et al., 2018).

The main contributions of our approach are: *(i)* the adoption of DatalogMTL to encode as rules a large class of temporal economic claims thus capturing their temporal logic, *(ii)* the rule synthesis according to our proposed *fine-tuned neural T5 model* (Raffel et al., 2020).

Our model is generated based on a dataset of manually crafted claim-query pairs inspired by economic newspapers and the Federal Reserve dataset (Fred, 2022). Finally, a validation phase attests the ability of our model to automatically synthesize newly generated claims unseen during fine-tuning.

2 Language and Templates

In this section we present the DatalogMTL formalism, the reference data model, and a categorization of claim-query pairs based on their semantics.

Temporal Datalog. We adopt *Datalog Metric Temporal Logic (DatalogMTL)*, a temporal logic-based language that has recently received great attention from the AI and database communities (Walega et al., 2019, 2020, 2021). DatalogMTL extends Datalog with metric temporal logic operators over the rational timeline. We consider DatalogMTL rules of the form $\bot := A_1, \ldots, A_k$, for $k \geq 0$, where all A_i are literals that follow the grammar $A ::= \top \mid \bot \mid P(\boldsymbol{\tau}) \mid \boxminus_\varrho A \mid \diamondminus_\varrho A$, with ϱ being a *non-negative* interval, and P an atom (on a predicate having either functional or infix notation) over a tuple of terms (i.e., variables and constants) $\boldsymbol{\tau}$. The conjunction of A is the rule *body* and commas denote the logical and ($\wedge$).

Given a database D, the semantics of a DatalogMTL rule is defined through interpretations. An interpretation $\mathfrak{M}$ specifies for each time point t, whether a ground atom $P(\mathbf{a})$ from D is true at t, in which case we write $\mathfrak{M}, t \models P(\mathbf{a})@t$. The @ symbol denotes the time reference for an atom.

An interpretation that satisfies all atoms of the body triggers the rule. The box minus $\boxminus_\varrho$ operator checks if a ground atom is *continuously* true in the interval ϱ, that is, $\mathfrak{M}, t \models \boxminus_\varrho A$ iff $\mathfrak{M}, s \models A$ for all s, with $t - s \in \varrho$. The diamond minus $\diamondminus_\varrho$ operator checks if a ground atom is true at least once in the interval ϱ, that is, $\mathfrak{M}, t \models \diamondminus_\varrho A$ iff $\mathfrak{M}, s \models A$ for some s, with $t - s \in \varrho$.

Data Model. We rely on existing data sources containing time series, each representing an economic metric and following the *key-value* format, where *key* represents a time instant and *value* the corresponding measure. Time series with multiple attributes can also be treated after a pre-processing

phase which restructures them into multiple tables having replicated key values. Instances of our data model straightforwardly derive from widely adopted on-line economic libraries such as the Federal Reserve Economic Data (FRED) and the datasource exposed by the central bank in our project.

Query templates. We distinguish temporal claims according to their semantics and organize them into templates, providing for each of them the corresponding DatalogMTL rule. Our rules can be efficiently executed on a DatalogMTL engine (Bellomarini et al., 2021; Benedikt et al., 2017).

In Table 1, we collect 10 different templates along with a claim/rule example in our scenario. It is not our intention to produce an exhaustive list of all the admissible templates, instead we aim at showing the feasibility of generating the correct rules for claims falling into any of the given input set of templates. Among others, the rule for Template 2 is triggered if the inflation during 2011 surpassed 2%. Our claims do not specify the country, as we refer to one single national central bank.

3 Translating Claims to DatalogMTL

In the context of Neural Machine Translation (Sutskever et al., 2014; Bahdanau et al., 2015) pre-trained Language Models (PLMs) neural networks based on Transformers excel at handling Natural Language (NL) sequences in understanding and generation tasks (Vaswani et al., 2017; Devlin et al., 2019). These models are typically pre-trained on large text corpora in an unsupervised manner, for example by predicting the missing tokens in each sentence. Many Natural Language Processing (NLP) tasks can benefit from the NL understanding and generation capabilities of PLMs by means of *fine-tuning* with task-specific data through transfer learning. Fine-tuning is performed in a supervised manner, by providing the labelled input.

Our approach fine-tunes the T5 model to use its text-to-text capabilities for our text-to-rule translation task. To this end, we generate and augment a fine-tuning set of textual input (claims) and output (rules) labels to specialize the T5 model.

Starting from a template-based categorization of the rules (Table 1), we transform each pair ($\langle$claim *NL*, rule *DL*$\rangle$) into a full-text specification: *NL* is prefixed by *"datalog translation:"*, which identifies the downstream translation task; for *DL* we safely remove the symbol $\perp$ appearing in all queries (thus non necessary to predict) and replace math symbols with text snippets.

We exploit the set of templates and the time series to augment the fine-tuning dataset. For each claim, we create a set of variants having the same semantics but different syntax. We replace the actual values of economic metrics, events, numeric values, temporal and comparison text snippets with specific placeholders. We show the process for claims belonging to templates 3 (1,2) e 2 (3,4):

```
1: In <instant> <metric> reached the
<max-min> since the last <epoch>.
2: During the last <epoch> <metric> is
<higher-lower> than in <instant>.
3: <metric> during <instant> was
<higher-lower> than <value>.
4: The value of <metric> was
<higher-lower> than <value> in <instant>.
```

We replace their rules with placeholders:

```
1-2: Time op. star[<epoch>] <metric>(y),
<metric>(x)@[<instant>],x <comparator> y
3-4: <metric>(x)@[<instant>],x
<comparator> <value>
```

We then replace the placeholders with the actual input values from our data sources:

```
1: In 2020 inflation reached the maximum
since the last six years.
2: During the last six years inflation
is lower than in 2020.
3: Inflation during 2011 was higher
than 0.02.
4: The value of inflation was higher
than 0.02 in 2011.
```

We apply the same process to the rules:

```
1-2: Time op. star[six years] inflation
(y), inflation(x)@[2020],x less than y
3-4: inflation(x)@[2011],x less than
equal to 0.02
```

4 Training and Validation Results

We describe: *(i)* how we obtained the fine-tuned T5 model and *(ii)* how we used it to automatically translate textual claims into rules. Our results are based on the exact string matches accuracy of predicted DatalogMTL rules against the ground truth. Traditional BLEU and ROUGE metrics are not used here since they may lead to good accuracy also for minimal variations of the predicted rules resulting to drastically different execution outcomes.

Model fine-tuning. Following our translation approach we generated 60K $\langle NL, DL\rangle$ pairs uniformly distributed among the 10 templates after undersampling the most populated classes. After splitting the dataset into training and test sets and setting the architecture parameters, we fine-tune a T5-base

#	Template	Claim / Rule Example
1	Metric at two time instants	In 2020 the inflation was higher than in year 2021 $\perp := \textit{inflation}(x)@[2020], \textit{inflation}(y)@[2021], x \leq y$
2	Metric value at single instant	Inflation during 2011 was higher than 0.02 $\perp := \textit{inflation}(x)@[2011], x \leq 0.02$
3	Metric at single instant w.r.t. previous epoch value	In 2020 inflation reached the maximum since the last six years $\perp := \Diamond_{[0,6]} \textit{inflation}(y), \textit{inflation}(x)@[2020], x < y$
4	Metric at a given epoch	During the last ten years the inflation has always been higher than 0.02 $\perp := \Diamond_{[0,10]} \textit{inflation}(x), x \leq 0.02, \textit{true}@[\textit{today}]$
5	Metric at given epoch implies metric at single instant	After ten years of inflation above 2%, public debt surpassed 3000 $\perp := \textit{public debt}(x), x \leq 3000, \boxminus_{[0,10]} \textit{inflationGt}(y), y = 0.02$
6	Event at a single instant	An economic crisis occurred in 2008 $\perp := \textit{economic crisis}(x)@[2008], x = \textit{False}$
7	Event at a given epoch	A salary increment has been observed during the last 3 years $\perp := \boxminus_{[0,3]} \textit{salary increment}(x), x = \textit{False}, \textit{true}@[\textit{today}]$
8	Metric at given epoch implies required event	If inflation for the last ten years was higher than 0.02, we have a salary increment $\perp := \textit{salary increment}(x), x = \textit{False}, \boxminus_{[0,10]} \textit{inflationGt}(z), z = 0.02$
9	Event at given epoch implies metric at a single instant	Till to 5 years after a stock market crash, inflation remained below 0.02 $\perp := \textit{inflation}(x), x \geq 0.02, \Diamond_{[0,5]} \textit{stock market crash}(z), z = \textit{True}$
10	Event at epoch and metric at instant imply required event	In the 3 years after a salary increment, if inflation is lower than 0.02, there is a salary increment $\perp := \textit{salary increment}(y), y = \textit{False}, \Diamond_{[0,3]} \textit{salary increment}(z), z = \textit{True}, \textit{inflation}(x), x < 0.02$

Table 1: Query templates with the respective claim and temporal Datalog example.

model (Google, 2022) to generate our prediction model. This process has been carried out in 3 hours on a NC12s_v2 machine with 12 CPUs, 224GB RAM and a P100 Nvidia GPU.

Model prediction. We apply our fine-tuned model on newly generated input claims in any of our 10 templates. For each claim, we generate a set of variants sharing the same semantics but with different syntax by using 3 paraphrasing tools from the NL-Augmenter Python framework (Goyal and Durrett, 2020; Dopierre et al., 2021; Kumar et al., 2019). We generate 30 variants per claim after replacing placeholder variables with a set of reference values. We then place back placeholders and eliminate duplicated and erroneous claims, i.e., those that have a different set of placeholders w.r.t. the starting claim. Finally, placeholders are replaced with actual values before randomly selecting 5000 claims balanced across template categories. Given such newly generated test set (average Jaccard distance from the training set was 0.317), we predict the DatalogMTL rules with our model. The model correctly predicts (in less than one hour) 4495 over 5000 claims resulting in a 0.90 average accuracy.

Table 2 provides accuracy results about templates, each characterized by 4 features: number of rule placeholders, epoch vs instant-based placeholders, number of comparison operators, and claim type, i.e., a simple statement or a more complex conditional phrase. Results show decreasing prediction performance with claims of higher complexity, corresponding to rules with mixed time placeholders and comparison operators on metrics.

Template	# Plcs	Time	# Ops	Type	Accuracy
1	5	I+I	1	S	0.794
2	4	I	1	S	0.934
3	5	I+E	1	S	0.636
4	4	E	1	S	0.970
5	7	E	2	C	0.802
6	2	I	0	S	0.986
7	2	E	0	S	0.978
8	5	E	1	C	0.974
9	5	E	1	C	0.984
10	6	E	1	C	0.932

Table 2: Prediction accuracy for claims grouped by template. *# Plcs* denotes the # of rule placeholders, *Time* reports instant (*I*) and epoch (*E*) placeholders, *# Ops* is the # of comparison operators on metrics, *Type* denotes conditional (*C*) and simple (*S*) phrases.

5 Conclusion

This work shows that numerical temporal claims can be computationally verified exploiting time series. While preliminary results are promising, one major challenge is the lack of training data for the rule generation module. Templates are an effective solution, but the coverage of the system depends on the number of deployed templates. We are also working on how to create data explanations for true claims, as now we only show counter-examples. One road is to identify examples by perturbing the Datalog rule, as studied for SQL queries (Wu et al., 2017). We are looking at how to improve the validation step by comparing the effects of the generated rules, instead of their syntactical equivalence and by testing a real-world benchmark of claims.

Acknowledgments. We thank Livia Blasi, Markus Nissl, and Mohammed Saeed for their help.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Luigi Bellomarini, Markus Nissl, and Emanuel Sallinger. 2021. Query evaluation in DatalogMTL - taming infinite query results. *CoRR*, abs/2109.10691.

Michael Benedikt, George Konstantinidis, Giansalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and Efthymia Tsamoura. 2017. Benchmarking the chase. In *PODS*, pages 37–52. ACM.

Sebastian Brandt, Elem Güzel Kalayci, Vladislav Ryzhikov, Guohui Xiao, and Michael Zakharyaschev. 2018. Querying log data with metric temporal logic. *J. Artif. Intell. Res.*, 62:829–877.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Thomas Dopierre, Christophe Gravier, and Wilfried Logerais. 2021. Protaugment: Unsupervised diverse short-texts paraphrasing for intent detection meta-learning. *CoRR*, abs/2105.12995.

Fred. 2022. FRED - federal reserve economic data. https://fred.stlouisfed.org/.

Google. 2022. T5-base model. https://huggingface.co/t5-base.

Tanya Goyal and Greg Durrett. 2020. Preordering for controlled paraphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 238–252, Online. Association for Computational Linguistics.

Lars Kai Hansen and Laura Rieger. 2019. Interpretability in intelligent systems - A new concept? In *Explainable AI*, volume 11700 of *Lecture Notes in Computer Science*, pages 41–49. Springer.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2021. A deep dive into deep learning approaches for text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2846–2851.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3609–3619. Association for Computational Linguistics.

Preslav Nakov, David P. A. Corney, Maram Hasanain, Firoj Alam, Tamer Elsayed, Alberto Barrón-Cedeño, Paolo Papotti, Shaden Shaar, and Giovanni Da San Martino. 2021. Automated fact-checking for assisting human fact-checkers. In *IJCAI*, pages 4551–4558. ijcai.org.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Mohammed Saeed and Paolo Papotti. 2021. Fact-checking statistical claims with tables. *IEEE Data Eng. Bull.*, 44(3):27–38.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems, NIPS 2014*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:6000–6010.

Przemyslaw Andrzej Walega, Bernardo Cuenca Grau, Mark Kaminski, and Egor V. Kostylev. 2020. Tractable fragments of datalog with metric temporal operators. In *IJCAI*, pages 1919–1925. ijcai.org.

Przemyslaw Andrzej Walega, Mark Kaminski, and Bernardo Cuenca Grau. 2019. Reasoning over streaming data in metric temporal datalog. In *AAAI*, pages 3092–3099. AAAI Press.

Przemyslaw Andrzej Walega, Michal Zawidzki, and Bernardo Cuenca Grau. 2021. Finitely materialisable datalog programs with metric temporal operators. In *KR*, pages 619–628.

You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2017. Computational fact checking through query perturbations. *ACM Trans. Database Syst.*, 42(1):4:1–4:41.

Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph. 2021. Neural machine translating from natural language to sparql. *Future Generation Computer Systems*, 117:510–519.