Legal Knowledge and Information Systems
E. Francesconi et al. (Eds.)

doi:10.3233/FAIA220453

A Multi-Step Approach in Translating Natural Language into Logical Formula

Ha-Thanh Nguyen [a] Fungwacharakorn Wachara [a] Fumihito Nishino [a] Ken Satoh [a]
[a] *National Institute of Informatics, Japan*
nguyenhathanh@nii.ac.jp

Abstract. Translating often has the meaning of converting from one human language to another. However, in a broader sense, it means transforming a message from one form of communication to another form. Logic is an important form of communication and the ability to translate natural language into logic is important in many different fields, in which logical reasoning and logical arguments are used. In the legal field, for example, judges must often reason from facts and arguments presented in natural language to logical conclusions. In this paper, toward the goal of support for this kind of reasoning with machines, we propose a method for translating natural language into logical representations using a combination of deep learning methods. Our approach contributes methodologies and insights to the development of computational methods for converting natural language into logical representations.

Keywords. deep translation, natural language, logical representation

1. Introduction

Although the law is written in natural language, this form is probably not the best option. First, natural language is ambiguous. This means that there can be more than one interpretation of what a sentence in law means [1]. This can lead to confusion and can make it hard to enforce the law. Second, natural language is constantly changing [2]. Words can change meaning over time, which can become a challenge in understanding old laws. Finally, natural language is not the same in every country. This can make it hard to write laws that can be enforced internationally. Representing the law in a logical form, such as first-order logic, can help to solve these problems. First-order logic is a formal language that can be used to write down laws in a way that is clear and unambiguous. First-order logic is also not subject to the same problems as natural language. This is because first-order logic is a fixed and well-defined language. This means that the meaning of the formula in first-order logic is the same in every country.

Using logic programming to support judges and attorneys has many benefits. Perhaps the most significant benefit is that it can help to ensure that the law is applied consistently across cases. By representing the law using formal logic, it becomes possible to develop an automated system that can check for inconsistencies and errors. As a result, the quality of legal decision-making can be improved. Another benefit of using logic programming to support judges and attorneys is that it can help to make the law more

accessible to the general public. By representing the law using formal logic, it becomes possible to develop systems that can generate plain-language explanations of the law. This can help to increase public understanding of the law and improve compliance with the law. Finally, using logic programming to support judges and attorneys can help to make the law more efficient. By representing the law using formal logic, it becomes possible to develop systems that can automatically generate legal documents. This can save significant time and effort for both judges and attorneys.

Despite the above benefits, writing logical formula is a huge challenge. Writing correct logical expressions for law sentences requires expertise and experience. No matter how powerful the system is, without correct inputs, the result can be incorrect. In addition, the result of the reasoning system might be overwhelming to users which are not familiar with logical reasoning. Therefore, using law expert systems that are based on logic programming is not a trivial task. The most difficult requirement is to design the system so that law experts, usually those who are not specialized in logic, can give input to the system. Efforts to date such as controlled natural language or existing translation systems have not met this requirement.

The ideal system should translate both legal rules and case descriptions written in natural language sentences into logical formulas so that judgement could be fully automatically made. However, as far as legal rule translation is concerned, it is usually very difficult to translate legal rules written in natural language into logical formulas. It is because there are some hidden conditions only derived from legal interpretations of the whole set of legal rules and this information only can be obtained from legal literature, not from legal rules themselves. We, therefore, take the intermediate approach for producing judgment as follows.

1. We manually translate legal rules into logical formulas with the consultation of legal scholars and legal literature.
2. We translate case descriptions into logical formulas (fact formulas) automatically.
3. We reason about judgement by applying logical rule formulas to logical fact formulas.

For step 2, the problem is how to extract relevant information from case descriptions. One approach would be pattern-based information retrieval from case descriptions [3]. However, their approach is not robust, that is, if sentences do not follow pre-designed patterns, we cannot extract information, and preparing for various patterns manually will be very time-consuming.

In this paper, we take a robust approach to retrieve relevant fact formulas from a case description written in natural language using large pre-trained language models. To do so, we need to answer the following two questions: First, we want to see whether a large language model trained on a large amount of natural language text can be used to automatically generate logical formulas that are consistent with the text after some finetuning. Second, we want to see the weakness of this approach by looking at the types of errors the model makes. To answer the first question, we use large pretrained language models to translate natural language into logical formulas. To answer the second question, we examine the types of errors the model makes. We find that large pretrained language models can be used to automatically generate logical formulas that are consistent with the text after some finetuning. However, with the translation approach, a few slight mistakes in the prediction results can cause syntax errors in logical formulas, which is

tolerable in natural language translation. This information can be used to prepare necessary resources and improvements for the model.

Our main contribution is a novel approach to reason about judgments using manually translated logical rule formulas and automatically translated logical fact formulas. In particular, for automatic translation from case description into logical facts, we combine translation and correction of natural language into a logical formula using deep learning. To this end, we develop an effective deep learning framework with an appropriate training strategy to perform the translation and correction with PROLEG syntax[4]. We conduct experiments to verify the effectiveness of our approach and discuss the insights we gain from the experiments. The structure of this paper is as follows. In Section 2, we have preliminaries covering the basics of the existing logical form in English and the current status of machine translation methods. Section 3 provides a description of the proposed multi-step translation approach. In Section 4, we show the experiments and evaluation of the proposed model. Finally, we conclude in Section 5 and discuss some future works.

2. Backgrounds

2.1. Controlled Natural Languages

Controlled natural languages have been long invented for reducing ambiguity in natural languages. Basically, they are natural languages with restrictions of vocabulary and grammar so that sentences in such languages can be translated into unique logical expressions.

Attempto Controlled English (ACE) [5] is one notable controlled natural language which has been used for representing laws and regulations [6]. Each sentence in ACE can be translated into a first-order expression. For instance, the sentence *Every household creates some garbage* can be translated into $\forall x[household(x) \rightarrow \exists y[garbage(y) \wedge create(x,y)]]$. To achieve this, ACE restricts its vocabulary and grammar, for example, every common noun must occur in a noun phrase with a quantifier. Hence, some acceptable sentences in natural English are unacceptable in ACE. For instance, the sentence *Every household should pay tax for garbage* is unacceptable in ACE because there are no qualifiers for *tax* and *garbage*. Furthermore, strict interpretation rules of ACE may lead to unintended interpretations. For instance, since a pronoun in ACE refers to the most recent noun, *its garbage* in the sentence *Every household should pay some tax for its garbage* is interpreted to *the tax's garbage*, which is counterintuitive. In addition, it is hard to link the complex forms of words to their simple form. For instance, it is hard to link *taxation* to *tax*. Hence, simple forms are preferred to complex forms in ACE.

Logical English [7] is one more recent controlled natural language which has been used for representing laws and regulations [8]. Each sentence in Logical English can be translated into a Prolog rule. In the same manner to ACE, Logical English restricts its vocabulary and grammar, for example, the first occurrence of a common noun phrase must begin with *a/an* and the repeated noun phrase in the same sentence must begin with *the*. It is reported in [8] that Logical English attempts to serve as a syntactic sugar to make logic programs understandable for lay persons; however, it is still hard to write readable Logical English sentences. Assistive tools for writing in Logical English are being developed.

2.2. Existing Machine Translation Methods

A dictionary-based approach to automated translation relies on a set of lexical items, each mapping a concept to a set of alternative meanings. These dictionary entries are written by human experts and are often based on the structure of the source language [9]. In many cases, however, the mapping of concepts to a set of alternative meanings is not one-to-one. In these cases, the dictionary-based approach may require a human expert to choose the appropriate meaning for each concept. Another weakness of this approach is that it can not deal with the variance in wording that is characteristic of natural language. It is impossible to write a dictionary that covers all of the possible ways that a concept can be expressed.

Template methods generate translated sentences following a linear, rule-based approach [10]. This approach is also known as a finite state machine. The template method consists of a set of rules for different sentence templates, which are instantiated to input a sentence and generate a sentence. The simplest form of sentence generation is a sequence of replacement rules. This simple form is not sufficient for real-world applications and data-driven approaches are proposed. A data-driven approach is built using a corpus of parallel sentences, which can be used to build either a translation rule or a translation model.

The statistical approach to machine translation is based on the hypothesis that the translation of a text is a function of the source text and its context. This approach overcomes the problem of dictionary-based and rule-based machine translation by using a statistical model that can deal with the complexity of natural languages. Learning from data, the statistical approach can automatically discover the latent rules that govern the relationship between the source text and the target text. As a result, this approach does not require the knowledge of linguistics or the development of dictionaries and rules. In the era of big data, the statistical approach has become the most popular approach to machine translation with the development of neural machine translation. Large language models like BERT [11], BART [12], T5 [13] and GPT-3 [14] have achieved state-of-the-art results in various natural language processing tasks, including machine translation.

Although the problem of machine translation has been studied for a long time, the studies that directly solve the problem of converting from natural language to logical expressions are quite limited. The main reason is the lack of a large-scale dataset and a method for end-to-end learning from input natural language to output logical expression. In the software industry, converting natural language to another form of software resource like database schema receives more attention than logical expressions. Therefore, it becomes easier to find a large-scale dataset for database schema [15]. The second reason is that to be meaningful, the logical formula should be well defined in a formal language containing definitions and rules. Building such a language consumes a lot of effort in both research and development. Fortunately, PROLEG [4] provides a language for representing logical expression in the legal domain. With that condition, this work can be considered the first step to solving the problem of converting natural language to logical expressions automatically. In the domain of law, current studies about translation are not about machine translation but about human translation. In their paper, Witt et al. [16] provide a detailed analysis of the translation problem in the context of legal texts and propose approaches for improving human translation of legal texts into logical formulas.

3. Method

3.1. General Framework

Solving the problem of translating natural language into a logical formula, we need to deal with two challenges. The first one is the limitation of training data. There is no available dataset that can be used to directly train a model for this task. Furthermore, the professional annotators who can provide high-quality data are also limited. The second challenge is that the logical syntax is much more rigid than the natural language. The translation should be very precise in order to get the correct logical formula. We find that this is an interesting and challenging problem that has not been well explored in the field of natural language processing and deep learning.

We propose the general framework as in Figure 1. There are two main modules: translation and correction. Both of them are pretrained language models. By the universal approximation theorem, all the functions can be approximated by a very deep neural network. Theoretically, we only need one neural network for translation. However, in the context of limited training data, we need to have an appropriate approach to make the system more robust. The translation module is responsible for the translation of the source sentence into the target sentence. The correction module is responsible for the final correction of the translation result. The training data for the translation module is required more expert knowledge and human effort to annotate, but the training data for the correction module is much easier to obtain. This is the key idea that makes our method more effective and efficient, making the goal feasible with limited expert annotations. We present in detail how to prepare the training data for translation and correction modules in Section 3.2.

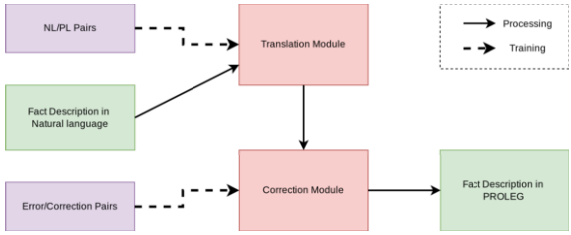

Figure 1. The general framework of the system.

3.2. Data Preparation

To prepare the data for the translation module, we have experts construct the pairs of fact descriptions in natural language and in logical representation (i.e. PROLEG). The logical representations need to be syntactically correct and follow the PROLEG grammar; however, the natural language descriptions can have multiple variations. As an initial investigation, we ask the experts to create logical representations for 15 different scenarios and their variance. Finally, we have 150 logical representations and their corresponding natural language descriptions. An example of data is shown in Table 1.

With a relatively small set of 150 samples, we do not expect that the translation module will be able to cover all the possible variations of the natural language descriptions.

Input
This Room Rental Agreement(this “Agreement”) is made and executed on the next business day, by and between Sanna Mirella Marin(hereinafter referred to as “Landlord”); and LEITCH Michael(hereinafter referred to as “Tenant”). In consideration whereby the Landlord leases the leased premises with the address located at Sunrise Village, Arkansas, 72207 United States. In consideration that the lease term shall commence on October 15, 2018 with the agreed payment term by which the tenant shall make the payment on a monthly basis. The Tenant shall pay the amount of $5 per month to the Landlord on an agreed payment basis.
Output
- Lessor(“Sanna Mirella Marin”). - Lessee(“LEITCH Michael”). - agreement_of_lease_contract(“Sanna Mirella Marin”,“LEITCH Michael”,“Room”,“Sunrise Village, Arkansas, 72207 United States”,“$5 per month”,“2018 year 10 month 15 day”,_, “the next business day”,“This Room Rental Agreement”).

Table 1. Sample of data for translation module.

Strategy	Original	Error
Random capitalization	This is a sample sentence	this is a sAMple sentence
Random split	This is a sample sentence	Th is is a sample sent ence
Random removing	This is a sample sentence	This is a smple setence
Random adding	This is a sample sentence	This is a saample senjtence
Random replacing	This is a sample sentence	This is ant sample sentence

Table 2. Example of error generation strategies.

Interestingly, its errors follow some patterns, which can be learned by another neural network module (the correction module). Data preparation for the correction module can be done in several automatic strategies with examples described in Table 2. We generate a dataset of 20,000 samples, which is much larger than the training set of the translation module.

3.3. Training

In the framework described in Figure 1, the translation module and correction module are trained in a sequential manner:

- Train the translation module with the 150 samples constructed in Section 3.2.
- Train the correction module with the generated incorrect logical representations.
- Use the trained translation module and correction module to translate new natural language descriptions into logical representations.

We do not train the models from scratch but make use of large version of BART to initialize the parameters of our models. We train the translation module and the correction module with early stopped by validation set.

4. Experiments

4.1. Evaluation Metrics

We implement two metrics for this typical translation task. The first metric is exact match accuracy and the second is the longest common non-continuous subsequence (LCNS). We do not use BLEU [17] or ROUGE [18] or other standard translation metrics because these are not suitable for the problem of translating natural language to logical formulas. With these metrics, there need to be several reference translations from which the translation can be judged. In our case, we have only one reference translation. The syntax of logical formulas is very rigid and they do not accept variance in the translation. The exact match accuracy can let us know the percentage of translation units that are exactly matched with the reference translation (ready for the application) and the LCNS can tell how far the current performance is from the ideal case. The exact match evaluation is calculated by Formula 1.

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}_{pred_i=gold_i} \tag{1}$$

To compute the LCNS evaluation, we implement a dynamic algorithm to find the longest common non-continuous substring of two strings *pred* and *gold* as in Algorithm 1:

Algorithm 1 LCNS's algorithm

procedure LCNS($gold, pred$) ▷ lcns of gold and pred
  $m \leftarrow$ `length of` $gold$
  $n \leftarrow$ `length of` $pred$
  L `size of` mxn ▷ dynamic programming matrix
  for $i \leftarrow 1,2,...,m$ `and` $j \leftarrow 1,2,...,n$ **, do**
    if $pred_i = gold_j$ **then**
      $L_{i,j} \leftarrow L_{i-1,j-1} + 1$
    else
      $L_{i,j} \leftarrow \max\{L_{i-1,j}, L_{i,j-1}\}$.
    end if
  end for
  return $L_{m,n}$ ▷ The lcns is $L_{m,n}$
end procedure

The LCNS evaluation value is the division of LCNS($gold, pred$) by the length of $gold$.

4.2. Experimental Results

Table 3 shows the experimental result of the translation system with and without the correction module. With a relatively small dataset, the translation module itself can produce a translation that has about 45% of exact match accuracy with the reference translation

Table 3. The experimental result of the translation system with and without correction module

System	Exact Match	LCNS
Translation Module w.o. Correction Module	0.45	0.94
Translation Module w. Correction Module	0.58	0.97

and 94% of LCNS. The correction module can further improve the translation quality to 58% of exact match accuracy and 97% of LCNS. We can have three observations from these experimental results:

- The translation module alone can produce a translation with fairly high quality.
- The correction module can further improve the translation quality. The improvement is not that significant given the small training dataset, but it is significant enough to show that the correction module has the potential to improve the translation quality.
- The translation quality can be further improved if we have a larger training dataset.

While the Exact Match and LCNS metrics show us the overall performance and improvement of the system, it's hard to imagine how good the system really is and what errors the correction module can detect and correct. Therefore, in Section 4.3, we analyze the specific cases that we observed in our experiment.

4.3. Error Analysis

Table 4. Translation examples for the translation system with and without correction module

	Expected/Generated Formula	LCNS
Gold Translation	fact_of_duress(personC,personB,rescission(contract0),1998 year 5 month 25 day).	-
w.o Correction Module	fact_of_duress(personC,personB,resCission(Contract0),1998 year 5 month 25 day).	0.97
Full System	fact_of_duress(personC,personB,rescission(contract0),1998 year 5 month 25 day).	1.00
Gold Translation	manifestation_fact(rescission(contract0),personA,personB,2030 year 12 month 1 day).	-
w.o Correction Module	manifeStation_fact(resCission(Contract0),personA,personB,2030 year 12 month 1 day).	0.96
Full System	manifestation_fact(rescission(contract0),personA,personB,2030 year 12 month 1 day).	1.00
Gold Translation	contract(personB,personA,this_real_estate,300,1995 year 11 month 13 day,contract0).	-
w.o Correction Module	contract(personB,personA,this_real_ estate,300,2000 year 11 month 13 day	0.81
Full System	contract(personB,personA,this_real_estate,300,2000 year 11 month 13 day	0.81

Table 4 shows some translation examples for the translation system with and without the correction module. The reference translation is provided in the first column. The translation generated by the translation module is in the second column and the translation generated by the translation module with the correction module is in the third column. We can see that the system can extract the key information from the input sentence and generate a translation with fairly high quality. Although the predicates in the target logical form do not appear in the input sentence, the system can produce a correct translation by reusing the existing predicates in the training dataset. In the version with solely the translation module, the system sometimes misspells the predicates and variable names in the logical formula. The correction module can correct some of these errors. In some cases, the correction module can also correct the case of the predicate names and variable names, which is important for the correctness of the logical formula. In the case of wrong date translation, the correction module based on noised generation can only reformat the date but can not assure the correctness of the date.

Looking at Table 4, we see that in the first two examples, the translation module produces pretty good but not perfect outputs. Words like “rescission”, “contract” are miscapitalized as “resCission”, “Contract”. These errors are insignificant to humans, and may not be detected at a glance. The LCNS metric also shows that these translations are almost identical to the original (0.96-0.97). However, they are serious errors for the reasoning engine, which is very strict in syntax. Correction module is effective in cases like this.

However, in the last example, where the translation module completely distorts the content of the argument, the correction module loses its effect. As we see, it can only remove an redundant space in the variable “this_real_ estate” and this is of no value in terms of formula correction. The logical expression is still both syntactically and semantically wrong. These analyzes give us suggestions for the design of models focusing on recognizing the arguments, which will be implemented and introduced in future work.

4.4. Discussions

From the experimental results, we can see that the translation system can achieve fairly high quality with a small amount of training data. However, the system still has some limitations. First, in our current dataset, the number of predicates and variables is limited. Therefore, the translation system can not generate a logical formula containing predicates and variables that are very different from the predicates and variables in the training dataset. Second, the translation system is not robust in recognizing the dates. In the third output in Table 4, the translation system translates the date “1995 year 11 month 13 day” to “2000 year 11 month 13 day”. This result should be due to the lack of date recognition skill in the model and overfit to the training dataset. The correction module can not deal with this kind of error. Third, the system sometimes returns an output that is not complete. Also in the third output, the function “contract” does not have all required arguments and the closing “)” is missing. In future work, the combination of template-based fill-in-slot training is a promising method to improve the completeness of the output. In addition, a larger dataset and more types of predicates and variables is required to improve the robustness of the translation system.

5. Conclusions

In this work, we propose a translation system that translates natural language text into logical formulas. We use PROLEG syntax as our target language. As an investigation step, we hire experts to create a small translation dataset from natural language to logical formulas. To improve the translation performance, we propose the multi-step framework by appending the correction module to deal with translationś errors. The correction module is constructed by learning a reverse function from the error generator. The experimental results show that this design can improve the translation quality. The translation system with the correction module can achieve 58% of exact match accuracy and 97% of LCNS in the current small dataset. This work is a promising first step toward automatically translating natural language into logical formulas. To make the system more robust, we need to either improve the quality of the data used to train the model or use a more sophisticated approach to handle the errors. In future works, we want to experiment with other approaches inspired by other work on neural machine translation.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers, JP17H06103, JP19H05470 and JP22H00543 and JST, AIP Trilateral AI Research, Grant Number JPMJCR20G4, Japan.

References

[1] Allen LE, Lysaght LJ. Modern logic as a tool for remedying ambiguities in legal documents and analyzing the structure of legal documents' contained definitions. In: Logic in the Theory and Practice of Lawmaking. Springer; 2015. p. 383-407.
[2] Christiansen MH, Kirby S. Language evolution. OUP Oxford; 2003.
[3] Navas-Loro M, Satoh K, Rodríguez-Doncel V. ContractFrames: bridging the gap between natural language and logics in contract law. In: JSAI International Symposium on Artificial Intelligence. Springer; 2018. p. 101-14.
[4] Satoh K, Asai K, Kogawa T, Kubota M, Nakamura M, Nishigai Y, et al. PROLEG: an implementation of the presupposed ultimate fact theory of Japanese civil code by PROLOG technology. In: JSAI international symposium on artificial intelligence. Springer; 2010. p. 153-64.
[5] Fuchs NE, Kaljurand K, Kuhn T. Attempto controlled english for knowledge representation. In: Reasoning web. Springer; 2008. p. 104-24.
[6] Wyner A. From the language of legislation to executable logic programs. In: Logic in the theory and practice of lawmaking. Springer; 2015. p. 409-34.
[7] Kowalski R. Logical english. Proceedings of Logic and Practice of Programming (LPOP). 2020.
[8] Kowalski B, Dávila J, CA CL, Calejo M. Logical English as a Programming Language for the Law. In: Programming Languages and the Law 2022; 2022. .
[9] Neff MS, McCord MC. Acquiring lexical data from machine-readable dictionary resources for machine translation. Citeseer; 1990.
[10] Och FJ, Ney H. The alignment template approach to statistical machine translation. Computational linguistics. 2004;30(4):417-49.
[11] Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805. 2018.
[12] Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:191013461. 2019.
[13] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. J Mach Learn Res. 2020;21(140):1-67.
[14] Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language models are few-shot learners. Advances in neural information processing systems. 2020;33:1877-901.
[15] Ahkouk K, Machkour M, Majhadi K, Mama R. SQLSketch: Generating SQL Queries using a sketch-based approach. Journal of Intelligent & Fuzzy Systems. 2021;40.
[16] Witt A, Huggins A, Governatori G, Buckley J. Converting copyright legislation into machine-executable code: interpretation, coding validation and legal alignment. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law; 2021. p. 139-48.
[17] Papineni K, Roukos S, Ward T, Zhu WJ. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics; 2002. p. 311-8.
[18] Lin CY. Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out; 2004. p. 74-81.